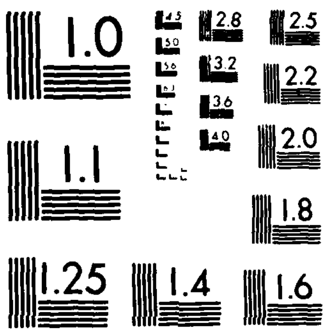
AD-A171 873 A SYSTEM FOR INTERACTIVE COMPUTER CONTROL OF EXPERIMENTS(U) NAVAL RESEARCH LAB WASHINGTON DC T A HARGREAVES ET AL. 25 AUG 86 NRL-MR-5743 1/2

UNCLASSIFIED F/G 9/2 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

Naval Research Laboratory

Washington, DC 20375-5000 NRL Memorandum Report 5743 August 25, 1986

(2)

AD-A171 873

A System for Interactive Computer Control of Experiments

T. A. HARGREAVES* AND M. E. READ

High Power Electromagnetic Radiation Branch
Plasma Physics Division

**JAYCOR, Inc.*
Alexandria, VA 22304

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS A171872
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) NRL Memorandum Report 5743	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Research Laboratory 6b. OFFICE SYMBOL (If applicable) Code 4740	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Washington, DC 20375-5000	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Department of Energy 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code) Washington, DC 20585	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO. DN780-307

11. TITLE (Include Security Classification)
A System for Interactive Computer Control of Experiments

12. PERSONAL AUTHOR(S)
Hargreaves, T. A.* and Read, M. E.

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Interim	FROM 9/83 TO 9/85	1986 August 25	136

16. SUPPLEMENTARY NOTATION
*JAYCOR, Inc., Alexandria, VA 22304

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Data acquisition PC

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

An integrated system combining both hardware and software has been developed to aid in the control and data acquisition of current experiments. The software is written in such a way as to be generally usable in a wide variety of experiments with little or no modification. The system is based on an IBM PC-XT and a CAMAC crate that can be filled with differing modules dependent upon experimental need.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Michael E. Read	22b. TELEPHONE (Include Area Code) (202) 767-4004 22c. OFFICE SYMBOL Code 4740

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

CONTENTS

A SYSTEM FOR INTERACTIVE COMPUTER CONTROL OF EXPERIMENTS

Introduction

An integrated system allowing computer control and data acquisition has been developed for the NRL quasi-optical gyrotron experiment. The motivation for this system is manyfold. First, it is necessary to simultaneously monitor several parameters; a difficult task using only oscilloscopes but simple for a computer. Secondly, a computerized system is potentially much more accurate than an experimenter reading numbers off of an oscilloscope and recording them in a notebook. The use of the computer allows the experimenter to visually monitor only the parameters of immediate interest while the computer quietly records all of the necessary data for future analysis. Additionally, since the data is already in the computer, data analysis and plotting is transformed into a process that takes a few minutes instead of large fractions of hours. Because much of this analysis can be done while the experiment is running, the taking of useless data can be held to a minimum. These facts all result in the ability to take much more complete and accurate data in a shorter time period than otherwise would be possible.

The quasi-optical gyrotron experiment for which this system was developed is a pulsed, repeatable experiment. Thus provision has been made to average the data over a number of different shots. Data to be taken falls into two broad categories. First is the data for which the entire wave form is desired, requiring a transient digitizer for each channel. Pulse lengths vary between 1 and 30 microseconds, so the software was written to digitize only the necessary time interval in order to avoid recording useless data during the shorter pulses. An example of this type of data is the output microwave pulse shape of the experiment. The second type of data to be taken requires only a single data point to be recorded each pulse. This is appropriate for parameters that do not change during the pulse such as the cavity magnetic field. Since this type of data requires very little storage space, virtually every experimental parameter can be measured and stored each pulse.

The software portion of this system was adapted from the PICAX program written by Robert Walraven. This provided the basic structure of the program, and only needed to be converted into a program capable of running on the IBM PC-XT. In addition, all of the experiment specific software such as the data taking, plotting, analysis, storage and experimental parameter control had to be developed. The users manual for the original PICAX, version

Manuscript approved December 16, 1985.

1, is attached, as is the manual and source listing for the current implementation, version 2. A short example of the program's usage is also attached.

Figure 1 shows a schematic of the hardware that is interfaced with the software. The IBM PC-XT™ is clearly the brains of the system, and communicates with the CAMAC crate through the crate controller. A wide variety of plug-ins is available for the crate, all of which can be controlled through the controller. The transient digitizers record the waveform data from the experiment while the digital to analog converter is used to control different pieces of the experimental apparatus. Additional capability is provided by the Data Translation DT2801 card which is used for its digital input and output as well as for recording the single point data from the experiment. To ensure that the data recorded all corresponds to the same point in time, each piece of single point data is held in a fast sample and hold circuit, and these are simultaneously triggered by a single trigger pulse. For accurate triggering and repetition rates an external pulse generator is used to trigger both the experiment and the diagnostics.

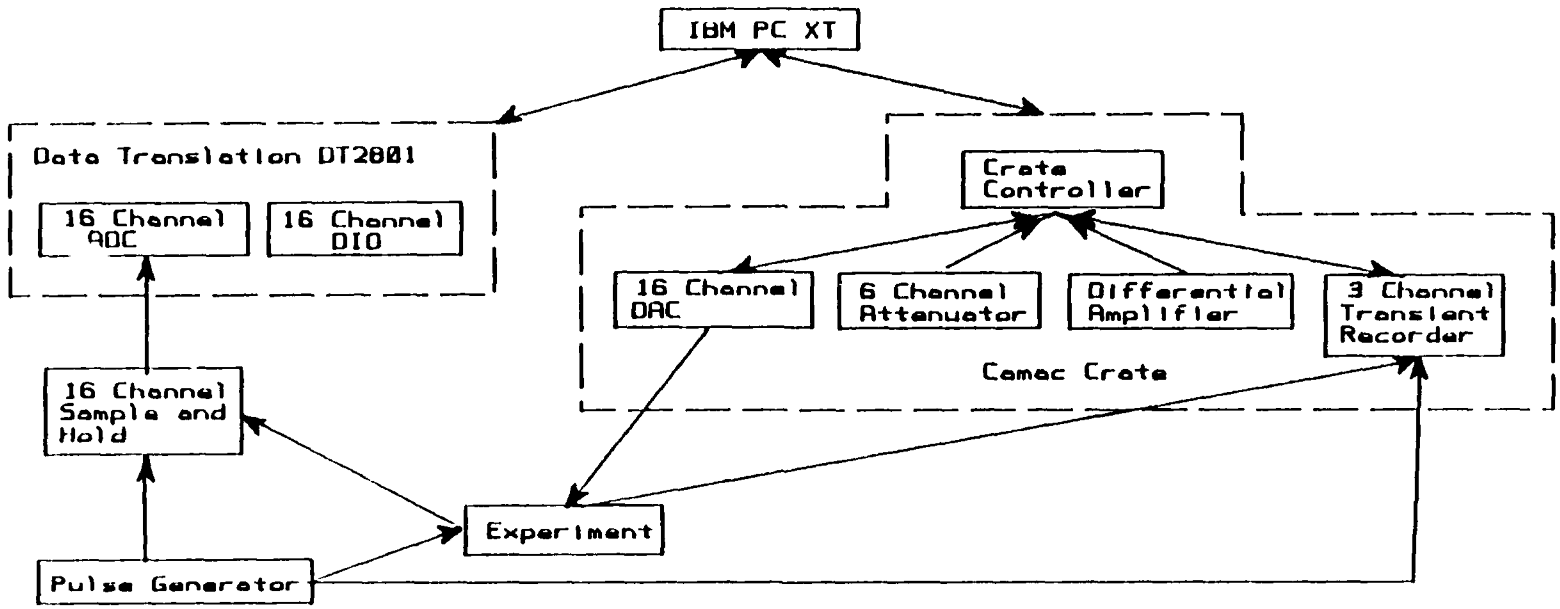


Fig. 1 — Data acquisition system

ATTACHMENT 1

PICAX

A Program for Interactive Control and Acquisition for eXperiments

USERS MANUAL

Robert Walraven
Department of Applied Science
University of California, Davis
3 December 1981
Program version 1.0

PREFACE

The PICAX program was born in the laboratory. An embryonic version, written in 1978, was used for control and data acquisition for an experiment in plasma physics. Soon this early version was modified to be used with other unrelated experiments; it satisfied many needs that seem to be universally desirable in any program for control and data acquisition, namely,

* Interactive control
* User-controllable data acquisition
* Storage and retrieval of data on mass storage
* On-line analysis of data
* Plotting of data and analyzed results

PICAX, the current version of that early program, is the result of three years of experience with the needs of both casual and advanced users.

If you are a PICAX user, send me your name and address so that I can send you information about problems and improvements. Also, if you have any suggestions about how PICAX could be improved, please let me know.

Robert Walraven

TABLE OF CONTENTS

INTRODUCTION

PICAX is a FORTRAN program written for the PDP-11 computer family and the RT-11 operating system. It is specifically designed to minimize the user's effort in programming for applications in which the computer is used for laboratory control and data acquisition by providing the following features:

* Standard commands that call user-written subroutines to control experiments, analyze and plot data, and read and write disk files.

* Standard, simple, relatively crashproof commands to OPEN, FIND, and CLOSE disk data files.

* DIR command to produce directory listings on the console or line printer.

* Commands that may be executed individually as they are typed or as a program of several commands (with simple looping capabilities).

* User-callable analysis subroutines for linear and non-linear fitting, digital filtering, Fast Fourier Transforms, correlation, etc., available in PICLIB library.

* Segmented structure suitable for efficient overlaid programming.

* Query feature for detailed descriptions of system and user commands.

SYSTEM COMMANDS

To use PICAX, you must first write FORTRAN-callable subroutines to perform at least some of the following operations:

* Initialize data and experiment
* Define user variables
* Take data
* Analyze data
* Plot data and results
* Write data to disk
* Read data from disk
* Describe user commands

The user-written subroutines must have specific standard names, and must be put in object modules with specific names so that the linker can place them properly in the overlay program it creates. Details on writing the user subroutines are given later.

Each command may be followed by up to four floating point numbers in free field format separated by commas. These numbers are called COMMAND VARIABLES, and are passed to the user-written subroutines through labeled COMMON.

Commands are executed in the immediate mode simply by typing the command code followed by a list of optional parameter values. For example, typing

```
PLOT -1,1.235,1E5
```

will cause a user-written plot subroutine to be called with the four command variables set to -1., 1.235, 1E5, and 0.0.

User variables are provided to define parameters for the user subroutines that you don't want to specify every time a command is typed. For example, you may wish to have user variables to describe the extent of the x-axis displayed in a plot, or the value to output to a D/A converter at the start of a run.

PICAX recognizes commands by looking for a one or two letter code. For example, any of the commands

```
DI6
DIR 6
DIRECTORY 6
DIXIECUP 6
```

will produce a directory listing on the line printer.

BRIEF LIST OF COMMANDS

The standard commands supported by PICAX are

A	Add to a user variable
AN	Analyze data
CL	Close the disk output file
DI	Directory listing
DO	Do loop
E	Erase screen
EX	Exit program
FI	Find an old disk file for output
GO	Go to program line
HE	Help the user out
HC	Make hardcopy of screen
K	Kill program
LC	List commands
LP	List program
LV	List user variables
O	Open a new disk file for output
PA	Pause
PL	Plot data
PR	Proceed with experiment
R	Read a record from disk
S	Start experiment
T	Enter title
V	Set a user variable
W	Write a record to disk
WA	Wait in units of 10 ms.
Z	Zero or init data

If you type a command and PICAX cannot find a match with one of the above commands, it will call the user-written routine UMATCH to see if the user has defined any additional commands.

DESCRIPTION OF COMMANDS

Note: Square brackets [] are used to indicate optional parameters for a command. The brackets should not be included when the command is actually typed.

A N,R Adds the value R to user variable number N. For example,

```
A 5,-12.1
```

will add -12.1 to user variable 5.

AN Calls a user written routine named UANLYZ to analyze data. Up to four command variables may be specified.

CL [LU] Closes a disk file on logical unit LU if one is open for output. If LU is not specified, logical unit 2 is assumed.

DI [LU] Produces a directory listing of DK:. The listing is normally output to the console, but

```
DI 6
```

will produce a listing on the line printer.

DO N,M Do loop command for program mode. N specifies the line that ends the loop, and M specifies how many times to execute the loop. For example,

```
DO 30,5
```

means DO to line 30 five times.

E Erase screen (if console is a Tektronix terminal).

EX Exit the program (return to RT-11). PICAX disables control-C exiting so that the program is forced to perform any final housekeeping before stopping. Typing this command is the only way to get out of PICAX.

FI [LU] Find an old disk file for input on logical unit LU. PICAX prompts the user for a file name, finds the file, reads the first record, and rewinds the file. If LU is not specified, 3 is assumed. Up to 6 logical units may be open for input at one time.

GO [N] Begin the current program. If N is specified, the program is started at line N.

HE Help the user out by typing some useful information.

HC Make a hard copy of the screen.

K Kill the current program. Deletes all lines of the program.

LC List the valid defined commands, and give a brief description of each.

LP List the current program.

LV List the current values of the user variables and give a brief description of each variable.

O [LU][,N] Open a new disk file for output on logical unit LU. If LU is not specified, 2 is assumed. PICAX prompts the user for a file name, closes any open files on the specified logical unit, and opens the new file. N is an optional file size. If N>0, a file is opened N blocks long. If N=0, a file is opened that is equal in length to half of the largest free space on the specified device. If N=-1, a file is opened that is equal in length to the largest free space on the specified device. Up to 6 logical units may be open for output at one time.

PA Waits until a return is typed.

PL Calls a user-written routine named UPLOT to plot data. Up to four command variables may be specified.

PR Proceed with the experimental data acquisition after it has been interrupted by the user typing something. Up to four command variables may be specified.

R [LU] Read a record from logical unit LU by calling the user-written subroutine UREAD. If LU is not specified, 3 is assumed. If no file is open for input on the specified logical unit, an error occurs. Up to four command variables may be specified, but the first must be the LU.

S Start the experimental data acquisition by calling the user-written subroutine USTART. Up to four command variables may be specified.

T Enter title. PICAX will prompt the user for a line of text to be used as a title.

V N,R Set user variable number N equal to R. For example,

```
V3 12.1
```

sets user variable 3 to 12.1.

W [LU] Write a record to logical unit LU by calling the user-written subroutine UWRITE. If LU is not specified, 2 is assumed. If no file is open for output, an error occurs. Up to four command variables may

be specified, but the first must be the LU.

WA N Wait for N times 10 milliseconds before proceeding to the next command

Z Zero data by calling the user-written subroutine UZERO. Up to four command variables may be specified.

The user may add additional commands to this list by specifying them in the user-written routine UMATCH.

If you do not remember what a particular command does, type the command followed by a question mark. PICAX will respond by giving you information about the command.

PROGRAM MODE

Commands may be executed as they are typed or in a group as a program. If a line number is typed before the command, the command is not executed immediately, but is entered into a program buffer. The following rules apply to programs:

1. A program may be up to 50 lines in length.

2. Line numbers may be any number in the range 1 to 999

3. Line numbers do not have to be in consecutive order.

4. Lines do not have to be typed in order; PICAX will automatically order the lines numerically.

5. To change a line, simply retype the new line with the same line number.

6. To delete a line, type the line number only.

7. To delete the entire program, type the command K.

8. To list a program, type the command LP.

9. To run a program, type the command GO. To start a program at a particular line number that is not the first line number, type the command GO followed by the line number.

The two commands DO and GO allow simple looping possibilities. A typical program with looping might look like this:

```
10 O 7
20 V5 = 10
30 DO 60,5
40 S 5,7
50 W 7,10.1
60 A 5,1
70 CL 7
80 GO 10
```

Line number 10 opens a file on logical unit 7. Line 20 sets user variable 5 to 10. Line 30 causes lines 40 through 60 to be executed in a loop 5 times. Line 40 causes the user-written subroutine USTART to be called with the command variables set to 5.,7.,0.,0. Line 50 causes the user-written subroutine UWRITE to be called for logical unit 7 with the command variables set to 7.,10.1,0.,0. Line 60 adds 1. to user variable 5. Line 70 closes unit 7. Line 80 loops back to the beginning.

PICAX PROGRAM FLOW

The following steps describe the program flow through PICAX as it is running:

1. Initialize PICAX. Call UINIT.

2. Write a '*' to prompt user.

3. If a return (after a line of input) or a control-C was typed, go to 8.

4. If the experiment is on (EXPT ON = .TRUE.), call UPDATE.

5. If a program is not running, go to 3.

6. Execute the next line of the program.

7. If the program is still running, go to 3. Otherwise, go to 2.

8. Turn experiment off (EXPT ON = .FALSE.). Turn program off.

9. If no characters were typed, go to 2.

10. If a control-C was typed, write an acknowledgement and go to 2.

11. If the typed input was not intelligible, write an error message, and go to 2.

12. If only a line number was typed, delete that line from the program and go to 2.

13. If a recognizable command was not found, write an error message and go to 2.

14. If there was a line number, insert the line into the program and go to 2. Otherwise, execute the command immediately and go to 2.

The logical variable EXPT ON determines whether an experiment is running or not. The logical variable PROG ON determines whether a program is running. These two variables may be accessed through the following COMMON block:

```
COMMON /P FLAGS/ PROG ON, EXPT ON, QUERY
LOGICAL PROG ON, EXPT ON, QUERY
```

The commands PR and S set EXPT ON to true before jumping to their respective user-written subroutines UPROCD and USTART. The command GO sets PROG ON to true.

UPDATE

Once the experiment is turned on (EXPT ON = .TRUE.), it will stay on either until the user types a return or control-C, or until the user-written subroutine UPDATE turns the experiment off. The commands PR and S should only be used to get the experiment going. If the experiment does need servicing, UPDATE should do whatever is required, and if the experiment is over, EXPT ON should then be set to false. In any case, UPDATE should return so that PICAX can check for input. If there is no input, UPDATE will be called again immediately. If there is input, however, EXPT ON will be set false and the input will be processed by PICAX. The experiment can be continued by typing PR or restarted by typing S.

Some experiments must be serviced rapidly, so the user may not wish to return to PICAX until the experiment is done. In this case the experiment cannot be interrupted by keyboard input.

OVERLAY STRUCTURE

PICAX is too large to fit in memory without using an overlay structure; this is primarily due to the inclusion of the AG-II graphics package. The overlay structure has been carefully designed to optimize the interactive response on a floppy-disk system. The overlay structure of PICAX consists of a root module and four overlay regions. The file PICAX.COM is a command file that performs the overlay linkage. All modules beginning with "PICAX" contain PICAX subroutines. All modules beginning with "AG2" are part of the AG-II graphics system. The user-written routines should be placed in the modules USERR, USER1, USER2, and USER3 as follows:

```
USERR:  UPDATE, UPLOT
USER1:  UANLYZ,UINIT,UPROCD,UREAD,
        USTART,UVLIST,UWRITE,UZERO
USER2:  UMATCH, UCMNDS,UINFO
USER3:  QUANAL,QUPLOT,QUPROC,QUREAD,
        QUSTRT,QUWRIT,QUZERO
```

USERR is a root module, and should contain any routines associated with fast data taking. That is why UPDATE is put there. The subroutine UPLOT is also placed in the root segment because all AG-II graphics routines that it might call are in the overlay modules. The module USER1 contains user-written subroutines that respond to various commands. The module USER2 contains subroutines for defining any additional commands that the user wishes to supply. USER3 contains help information about the user-written commands.

Modules with the names USERR.SKL, USER1.SKL, USER2.SKL, and USER3.SKL contain skeleton versions of all the required user-written subroutines, so that you only have to write the subroutines that are required by your specific needs.

PICLIB

The module PICLIB.FOR contains a number of subroutines that are of interest in a data acquisition and analysis environment, and that may be called by the required user subroutines. The subroutines in PICLIB.FOR are

FILTER - A general non-recursive filter that may be used to perform lowpass, bandpass, highpass, and bandstop filtering of data.

KAISER - A subroutine required by FILTER.

BESIO - A subroutine required by FILTER that computes the zero-th order modified bessel function I(x).

H OF Z NR - Computes the transfer function of a non-recursive filter.

CURFIT - Performs a least-squares fit to a user-specified non-linear function.

FCHISQ - A subroutine required by CURFIT.

MATINV - A subroutine required by CURFIT that computes the inverse of a symmetric matrix.

FAST - Performs a fast fourier transform of data.

The documentation for these subroutines is provided in the source listings.

AG-II

The advanced graphics package (AG-II) is functionally equivalent to the Tektronix Plot 10 advanced graphics package. However, it has been written to run faster and to fit on an LSI-11 by making use of the RT-11 overlay structure. Sources and command files for preparing the AG-II overlay modules are available from Tektronix Plot-10 Software Marketing: contact Will Gallant at (503) 682-3411, ext. 3785. You may obtain a manual for the package from your local Tektronix office (Manual part no. 070-2244-00).

EXAMPLE USER SUBROUTINES

Let's look at how PICAX could be used in a typical laboratory situation. Suppose you want to write a program to control an experiment that is interfaced to a DRV11 16-bit digital input/output card at address 167770. Further, suppose you would like the program to do the following:

1. At the start of the experiment, load the output register with 0.

2. Turn on bit 15 of the output register. Suppose a positive transition of this bit is used to enable the external experiment.

3. Wait until the DRV11 status register (bit 7) goes high, indicating that data is ready. Then read the data.

4. Turn off bit 15 of the output register.

5. If 128 data points have been collected, stop the experiment. Otherwise increment the output register and go to step 2.

6. After data taking is over, allow the user to filter the data, plot it, and write raw data to disk.

7. Let the program read data from disk, so that the data can be analyzed at some later time.

We will define a labeled common to pass variables among the user subroutines:

```
COMMON /DEMO/ NDATA, DATA(128), NRUNS
```

where NDATA is the number of the next data point to be obtained from the experiment, DATA is an array to store the accumulated data, and NRUNS is the number of accumulated runs if new data is added to the data already in the array DATA.

The required user subroutines for this task are shown below. These subroutines interact with PICAX through labeled commons:

1. The command variables that follow a command may be picked up with
   ```
   COMMON /P C VAR/ N C VAR, CVAR(1)
   ```
   where CVAR(1) through CVAR(4) are up to four command variables that follow the command.

2. The PICAX logical variable EXPT ON must be set false in the user subroutine UPDATE when data taking is over. EXPT ON is referenced through

```
COMMON /P FLAGS/ PROG ON, EXPT ON, QUERY
LOGICAL          PROG ON, EXPT ON, QUERY
```

3. The title, if any, defined by the T command may be obtained from

```
COMMON /P TITLE/ LTITLE(36)
```

4. The user variables may be obtained from

```
COMMON /U VAR  / N U VAR, U VAR(1)
```

   where N U VAR is the total number of user variables. The user variables should be set to their initial values in user subroutine UINIT. For convenience, the correct dimension for UVAR need only appear in UINIT, since the linker will set the length of the labeled common UVAR equal to its longest occurance. Everywhere else UVAR may be dimensioned 1.

The example user subroutines contain extensive documentation that is applicable to their use in PICAX, so it is recommended that the user read through these subroutine listings carefully.

```
C***************************************************************
C
C     USERR.FOR:  PICAX user module for the root segment
C
C***************************************************************
C
      SUBROUTINE UPDATE
C
C-----User root routine to service experiment.  If the S command was
C-----typed with the first command variable non-zero, the new data is
C-----added to DATA, otherwise it replaces DATA.
C
      COMMON /DEMO   / N DATA, DATA(128), N RUNS
      COMMON /P C VAR/ N C VAR, C VAR(1)
      COMMON /P FLAGS/ PROG ON, EXPT ON, QUERY
      LOGICAL PROG ON, EXPT ON, QUERY
C
C-----Return to PICAX if no data available yet
      IF ((IPEEK("167770).AND."200).EQ.0) RETURN
      VALUE = IPEEK("167774)             !Get next data point
C-----If CVAR(1) is zero, store it in the data array.  Otherwise
C-----add it to the data array.
      IF (CVAR(1).EQ.0.) DATA (NDATA) = VALUE
      IF (CVAR(1).NE.0.) DATA (NDATA) = DATA (NDATA) + VALUE
      I = IPEEK("167772) .AND. "77777 !Turn bit 15 off
      CALL IPOKE("167772,I)
      IF (NDATA.EQ.128) GO TO 10         !Jump to 10 if done
      NDATA = NDATA + 1                  !Increment output register
      CALL IPOKE("167772,NDATA)
      CALL IPOKE("167772,NDATA.OR."100000)    !Turn bit 15 on
      RETURN
C
   10 EXPT ON = .FALSE.                  !Turn experiment off
      N RUNS = N RUNS + 1                !Increment run count
      RETURN
      END
```

```
C---------------------------------------------------------------------
      SUBROUTINE UPLOT
C
C-----User routine to plot data.  If a nonzero command variable is
C-----entered, the first and second user variables are used as the
C-----lower and upper limits of the y axis.
C
      COMMON /DEMO   / N DATA, DATA(128), N RUNS
      COMMON /P C VAR/ N C VAR, C VAR(1)
      COMMON /P TITLE/ LTITLE(36)
      COMMON /U VAR  / N U VAR, U VAR(1)
      DIMENSION X(4),Y(129)
      DATA X/-1.,128.,1.,1./
C
      CALL BINITT                 !Initialize AG-II graphics
      CALL ERASE                  !Erase screen
      CALL MOVABS (100,725)       !Move to top of graph
      CALL HTEXT (LTITLE)         !Print title, if any
      IF (CVAR(1).NE.0.)          !If command variable 1 is not zero,
     1        CALL DLIMY(UVAR(3),UVAR(4))     !set y-axis limits
      Y(1) = 128                  !Set number of pts in plot array
      DO 10 I=1,128               !Load data in plot array
   10 Y(I+1) = DATA(I)
      CALL CHECK (X,Y)            !Plot data
      CALL DSPLAY(X,Y)
      CALL MOVABS (0,750)         !Move to upper left of screen
      CALL ALFMOD                 !Enter alphanumeric mode
      RETURN
      END
```

```
C***************************************************************
C
C     USER1.FOR:  PICAX user module for overlay region 1
C
C***************************************************************
C
      SUBROUTINE UANLYZ
C
C     Pass data through a digital filter
C
      COMMON /DEMO   / N DATA, DATA(128), N RUNS
      COMMON /P C VAR/ N C VAR, C VAR(1)
      COMMON /U VAR  / N U VAR, U VAR(1)
C
      DIMENSION COEF(10), TEMP(10)
C
      IF (CVAR(1).NE.0. .OR. CVAR(2).NE.0.) GO TO 10
      FLOW = UVAR(1)                    !Get lower cutoff
      FHIGH= UVAR(2)                    !Get upper cutoff
      GO TO 20
   10 FLOW = CVAR(1)                    !Use command variables
      FHIGH= CVAR(2)                    !   for cutoffs
   20 IFLAG = 0
      CALL FILTER (DATA,DATA,128,FLOW,FHIGH,50.,COEF,TEMP,10,IFLAG)
      RETURN
      END
C-------------------------------------------------------------------
      SUBROUTINE UINIT
C
C     Initialize user data and variables
C
      COMMON /U VAR   / N U VAR, U VAR(4)
C
      CALL UZERO                        !Zero user data
      N U VAR = 4                       !Number of user variables
      U VAR (1) = 0.                    !Lower filter cutoff
      U VAR (2) = .5                    !Upper filter cutoff
      U VAR (3) = 0.                    !Lower limit of y-axis
      U VAR (4) = 0.                    !Upper limit of y-axis
      RETURN
      END
C-------------------------------------------------------------------
      SUBROUTINE UPROCD
C
C     In this example, no specific action need be taken on a proceed.
C
      RETURN
      END
```

```
C-----------------------------------------------------------------------
      SUBROUTINE UREAD (LUN)
      COMMON /DEMO   / NDATA, DATA(128),NRUNS
      COMMON /P TITLE/ LTITLE(36)
      COMMON /U VAR  / N U VAR, U VAR(1)
C
C     If LUN is negative, it is the first read for the file,
C     so just read in the title and user variables.
C
      IF (LUN.GT.0) GO TO 10
      LUN = -LUN
      READ (LUN,END=200) NUVAR,LTITLE
      READ (LUN,END=200) (U VAR(I),I=1,NUVAR)
      RETURN
   10 READ (LUN,END=200) NRUNS, DATA(128)
      RETURN
  200 WRITE (5,30)
   30 FORMAT(' End of file')
      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE USTART
C
C     Start the experiment, then return to PICAX.
C
      CALL IPOKE ("167772,0)                    !Clear output register
      CALL IPOKE ("167772,"100000)              !Flip bit 15 on
      CALL IPOKE ("167772,0)                    !   and off
      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE UVLIST
C
C     This subroutine passes information to PICAX about the user
C     variables by calling the PICAX subroutine DEFINE.  The first
C     argument is the variable number.  The second argument is
C     'I' or 'R' depending on whether the LV (list variables)
C     command should type an integer or real value for that variable.
C     The third argument is a descriptive string for the variable.
C
      CALL DEFINE (1,'R','Lower cutoff of filter')
      CALL DEFINE (2,'R','Upper cutoff of filter')
      CALL DEFINE (3,'I','Lower limit of y-axis')
      CALL DEFINE (4,'I','Upper limit of y-axis')
      RETURN
      END
```

```
C-----------------------------------------------------------------------
      SUBROUTINE UWRITE (LUN)
      COMMON /DEMO   / N DATA, DATA(128), N RUNS
      COMMON /P TITLE/ LTITLE(36)
      COMMON /U VAR  / N U VAR, U VAR(1)
C
C     If LUN is negative, it is the first write to the file, so just
C     write the title and user variables.
C
      IF (LUN.GT.0) GO TO 10
      LUN = -LUN
      WRITE (LUN,ERR=20,END=40) N U VAR, LTITLE
      WRITE (LUN,ERR=20,END=40) (U VAR(I),I=1,NUVAR)
      RETURN
   10 WRITE (LUN,ERR=20,END=40) N RUNS, DATA(128)
      RETURN
   20 WRITE (5,30)
   30 FORMAT(' Error on write')
      RETURN
   40 WRITE (5,50)
   50 FORMAT (' End of file')
      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE UZERO
C
C     Zero user data
C
      COMMON /DEMO   / N DATA, DATA(128), N RUNS
      N DATA = 1
      N RUNS = 0
      DO 10 I=1,128
10    DATA(I) = 0.
      RETURN
      END
```

```
C*********************************************************************
C
C     USER2.FOR:  PICAX user module for overlay region 2
C
C*********************************************************************
C
      SUBROUTINE UMATCH
C
C     The user may define additional commands beyond the standard
C     ones.  Two additional dummy commands UA and UB are added here
C     to illustrate how this is done.  Commands are defined by
C     calling the PICAX logical function MATCH as shown below.
C     The first argument is a command number (greater than 26).
C     The second argument is the two letter string that will cause
C     PICAX to call the command.  The third argument is a descriptive
C     string for the command.
C
      IF (MATCH(27,'UA','Dummy user command A')) RETURN
      IF (MATCH(28,'UB','Dummy user command B')) RETURN
      RETURN
      END
C---------------------------------------------------------------------
      SUBROUTINE UCMNDS (N)
C
C     This subroutine is called by PICAX when a user command is
C     typed.  N is the number of the command corresponding to the
C     number given in UMATCH.
C
      IF (N.EQ.27) WRITE (5,10)
      IF (N.EQ.28) WRITE (5,20)
   10 FORMAT(' Dummy user command A called')
   20 FORMAT(' Dummy user command B called')
      RETURN
      END
C---------------------------------------------------------------------
      SUBROUTINE UINFO (N)
C
C     This subroutine is called by PICAX when a user command is
C     followed by a question mark.  It should write out some
C     helpful information about the command.
C
      IF (N.EQ.27 .OR. N.EQ.28) WRITE (5,10)
   10 FORMAT(' This user command does not do anything.')
      RETURN
      END
```

```
C*********************************************************************
C
C     USER3.FOR:  PICAX user module for overlay region 3
C
C*********************************************************************
C
      SUBROUTINE QUANAL
      WRITE (5,10)
10    FORMAT(' Performs digital filtering of data.  If no command'/
     1 ' variables are specified, user variable 1 is used for the'/
     2 ' lower cutoff and user variable 2 is used for the upper'/
     3 ' cutoff.  If command variables are specified,  command'/
     4 ' variable 1 is used for the lower cutoff, and command'/
     5 ' variable 2 is used for the upper cutoff')
      RETURN
      END
C---------------------------------------------------------------------
      SUBROUTINE QUPLOT
      WRITE (5,10)
   10 FORMAT(' Plots the data.  If no command variable is given,'/
     1        ' automatic scaling of the y-axis is performed.  If a'/
     1 ' automatic scaling of the y-axis is performed.  If a'/
     2 ' non-zero command variable is specified, user variable 3'/
     3 ' is used for the lower limit of the y-axis, and user'/
     4 ' variable 4 is used for the upper limit of the y-axis.')
      RETURN
      END
C---------------------------------------------------------------------
      SUBROUTINE QUPROC
      WRITE (5,10)
   10 FORMAT(' Proceed with the experiment.')
      RETURN
      END
C---------------------------------------------------------------------
      SUBROUTINE QUREAD
      WRITE (5,10)
   10 FORMAT(' Read a data record from disk.')
      RETURN
      END
C---------------------------------------------------------------------
      SUBROUTINE QUSTRT
      WRITE (5,10)
   10 FORMAT (' Start data taking.  If no command variable is'/
     1 ' specified, the new data overwrites the contents of the'/
     2 ' data array.  If a non-zero command variable is specified,'/
     3 ' the new data is added to the contents of the data array.')
      RETURN
      END
```

```
C-----------------------------------------------------------------
      SUBROUTINE QUWRIT
      WRITE (5,10)
   10 FORMAT(' Write a data record to disk.')
      RETURN
      END
C-----------------------------------------------------------------
      SUBROUTINE QUZERO
      WRITE (5,10)
10    FORMAT(' Zero data array.')
      RETURN
      END
```

Building a PICAX Program

To create a PICAX program, follow these steps:

1. The objects for the overlay version of the Advanced Graphics Package should be on SY:. If they are not, create them and put them there. (See the Advanced Graphics Package README file.) If there is not sufficient room on SY:, and they must be located elsewhere, then modify the command file PICAXL.COM appropriately.

2. The objects for PICAX should be on DK:. If they are not, compile the source files and put the objects there. (PICAXR.FOR, PICAX1.FOR, PICAX2.FOR, PICAX3.FOR, DIREC.FOR, IOFILE.FOR)

3. Modify the skeleton user modules USERR.SKL, USER1.SKL, USER2.SKL, and USER3.SKL as needed for your application. Compile the resulting sources and put them on DK:.

4. Run the command file PICAXL.COM to link the program together. The resulting file is PICAX.SAV.

ATTACHMENT 2

PICAX

A Program for Interactive Control
and Acquisition for experiments

IBM PC-XT Version
Users Manual

Tom Hargreaves
JAYCOR
205 S. Whiting Street
Alexandria, VA 22304
September 10, 1985
Program Version 2.0

PREFACE TO VERSION 1.0

The PICAX program was born in the laboratory. An embryonic version, written in 1978, was used for control and data acquisition for an experiment in plasma physics. Soon this early version was modified to be used with other unrelated experiments; it satisfied many needs that seem to be universally desirable in any program for control and data acquisition, namely,

* Interactive control
* User-controllable data acquisition
* Storage and retrieval of data on mass storage
* On-line analysis of data
* Plotting of data and analyzed results

PICAX, the current version of that early program, is the result of three years of experience with the needs of both casual and advanced users.

If you are a PICAX user, send me your name and address so that I can send you information about problems and improvements. Also, if you have any suggestions about how PICAX could be improved, please let me know.

Robert Walraven

PREFACE TO VERSION 2.0

This version of the PICAX program has been modified from the original to run on the IBM PC-XT personal computer under the PC-DOS operating system. The user modules have also been writen for the NRL Quasi-optical Gyrotron Experiment, although they are sufficiently general that they should be of general use with little or no alteration. I would like to thank Robert Walraven for the free use of the original code, without which this would have been a much more time consuming project. He can be contacted at:

Dr. Robert Walraven
Department of Applied Science
University of California, Davis
Davis, California 95616

Tom Hargreaves

TABLE OF CONTENTS

INTRODUCTION

This version of PICAX is written in Microsoft FORTRAN for use on an IBM PC-XT using the PC-DOS operating system. Furthermore, the user subroutines have been written specifically for use on the NRL Quasi-optical Gyrotron experiment, however, the program should be useful in a wide variety of experiments with little or no modification. PICAX in its current state has the following features:

* Data acquisition.
* Data analysis.
* Data storage.
* Data plotting.
* On line help.
* No overlay structure.

The theory behind this implementation of PICAX is to measure all of the relevant experimental parameters simultaneously. Of course, care must be taken to ensure that the computer knows which channel of the data acquisition system corresponds to each parameter.

This manual is designed to be used in conjunction with the USERS MANUAL for PICAX version 1.0.

SYSTEM COMMANDS

The system commands are described in detail in the PICAX Users Manual for version 1.0. Here is a brief listing of the commands supported by PICAX:

A	Add to a user variable
AN	Analyze data
CL	Close the disk output file
*DI	Directory listing
DO	Do loop
E	Erase screen
EX	Exit program
FI	Find an old disk file for output
GO	Go to program line
HE	Help the user out
HC	Make hardcopy of screen
K	Kill program
LC	List commands
LP	List program
LV	List user variables
O	Open a new disk file for output
PA	Pause
PL	Plot data
PR	Proceed with experiment
R	Read a record from disk
S	Start experiment
T	Enter title
V	Set a user variable
W	Write a record to disk
WA	Wait in units of 10 ms.
Z	Zero or init data

* This command is not currently implemented in PICAX version 2.0.

Note that any command followed by a question mark and a return will cause the on line help package to print out detailed information covering the use of the command in question.

USER COMMANDS

There is currently only one user written command:

TR Trigger on/off

This command sets bit 0 of port 0 on the Data Translation card either high (5 volts) or low (0 volts) depending on the number that follows the command. This signal can then be used to trigger a relay in the trigger line to cut off the trigger pulse to the experiment.

USER CAPABILITIES

The user subroutines have been written to enable the user to perform a wide variety of tasks, the most important being the logging of experimental data. The intent is to measure various voltages of interest in a pulsed experiment, typically 1-30 microseconds in duration. There are two types of voltage data to be recorded. First are the channels for which the entire wave form is to be recorded, such as the cathode voltage or the collector current. Second are the channels for which a single point during the pulse is desired. An example of this type of data is the current in any of the magnetic field coils or the calorimeter voltage.

The second type of data is taken with the use of a multi-channel sample and hold unit fed into a Data Translation DT2801 multi-channel data acquisition card plugged into one of the expansion slots in the IBM PC-XT. The sample and hold unit can be triggered simultaneously so that the data that the computer retrieves all correspond to the same point in time. The DT2801 then reads the data one channel at a time from the sample and hold unit into the memory of the IBM PC-XT. A maximum of 16 channels have been developed for this type of data.

A Transiac 2008F transient digitizer is used to record the total waveform of the appropriate data. Currently there are three actual channels of hardware, but the software is set up for as many as 5 channels and can easily be expanded. During the experimental pulse the data is stored in the internal memory of each transient digitizer, which resides physically in a CAMAC crate. The data can then be read into the memory of the IBM PC-XT through an interface card in an expansion slot of the computer which is connected to a controller in the crate itself. This data transfer is set up to proceed in the DMA mode.

In addition to the transient digitizers, there is also a LeCroy 8102 six channel variable attenuator in the CAMAC crate. By telling the computer which data points are connected to each attenuator channel, the computer can calculate the correct voltages for the various data. The attenuator values are read by the computer through the crate controller each time an experimental run is started.

Besides the attenuators there is a single channel Transiac 1020 differential amplifier which can also be read by the computer through the crate controller. This allows amplification of weak signals to voltages appropriate to be measured by the various analog to digital converters.

Once the data is in the memory of the computer it is desirable to store it on the 10 MB hard disk of the IBM PC-XT. This is accomplished in the PICAX program by first opening a file for data storage. When the write data command is given, all information including data as well as user programable variables is written into the file. The data can then be transfered to floppy disks for future analysis. There are, of course, analagous routines for reading data from a file into the computer memory.

The data can be analyzed any time that it is in the memory of the computer, whether it was just taken from the experiment or read from a disk file. In this manner the experimental data can be taken and quickly stored, then analyzed at a later time. At present the peak microwave power as well as the efficiency can be derived from the data.

Another convenient feature of the PICAX program is the ability to immediately plot the experimental data. The first type of plot is simply the voltage versus time plot of one of the transient digitizers for either a single pulse or an average over a number of pulses. A second type of plot that is desirable is a graph of one parameter as a function of a second, for example, peak microwave power as a function of average magnetic field. The data presented in this type of plot will typically be averaged over many pulses.

Finally, it would be desirable to control various experimental parameters through the PICAX program. Only a small part of the software has been completed and some of the necessary hardware has been acquired. The Data Translation DT2801 has two channels of digital to analog converters as well as sixteen channels of digital input or output. One channel of the digital output has been programmed to respond to the trigger on/off command. Also available is a Transiac 3016 sixteen channel digital to analog conveter that resides in the CAMAC crate. Two of these channels have been programmed to set the superconducting magnet coil currents to the values specified by the user programmable variables. With this hardware combined with varying amounts of interface, it should be possible to control many more of the experimental parameters such as the electron gun magnetic field and the cathode voltage.

USER SUBROUTINE FLOW

The user subroutines are collected into the files USERR.QO, USER1.QO, USER2.QO, USER3.QO, USER4.QO, and USER5.QO. Each subroutine is well commented and the user is refered to the individual routine for more detailed information than is presented here. USERR.QO contains the plotting subroutines, and USER1.QO holds the initializing subroutine as well as the data analysis, disk file input and output routines and the data array zeroing routine. USER2.QO has the user command subroutines in addition to the user variable listing routine, while USER3.QO contains the online help package for the user written subroutines. USER4.QO holds both the routines to control the experiment as well as the routines to convert the raw data collected by the computer into real world units, and USER5.QO has all of the routines necessary for the actual data acquisition.

Immediately upon starting the PICAX program many variables need to be initialized. This is accomplished for the user written subroutines in the subroutine UINIT. Here such parameters as the memory address of the CAMAC crate controller are set. This subroutine is called only once, so that only parameters that will not change are set here.

To start the experiment and take data, the "S" command is used. This calls the subroutine USTART which sets up the computer to collect the data. The type of data to be acquired, number of shots to average over, etc. are all determined by the previously set USER VARIABLES. Data is then actually collected in subroutine UPDATE, which is called automatically. The raw data is first summed and then averaged here before being converted into real world units. After each data pulse, control is returned to PICAX to check for any keyboard input. If any input exists, the experiment is suspended and the input interpreted. The experiment may be resumed by using the PICAX command "PR" which calls the subroutine UPROCD. New attenuator values will be read before restarting the experiment, so care must be used when interpreting the results of such restarted experiments. Upon completion of the experiment and conversion of the data, program control is returned to PICAX.

The subroutines UPLOT, UREAD, and UWRITE are simply called by PICAX in response to the "P", "R", or "W" commands to either plot, read from a disk file or write the data to a disk file, respectively. The subroutine UANLYZ is called in when the "AN" command is executed and is designed to provide data analysis. UVLIST is the subroutine called by the command "LV" and will list the USER VARIABLES. These commands all execute a single function and then return complete control to PICAX.

BUILDING PICAX

To build the PICAX program, the following steps should be taken:

1) Collect all of the necessary source files onto the directory to be used. A list of the needed files can be found in the link file PICAX (see below).

2) Compile each of the source files. The FORTRAN files can be compiled by using the batch file FORT.BAT which calls the appropriate Microsoft Fortran compiler programs (see the listing below). To execute type:

   FORT (filename)

   The ASSEMBLY language routines must be compiled by the Microsoft Assembler MASM. At the completion of this step the source files are no longer needed, but the object files must all reside on the directory being used.

3) Link the object files together to produce the executable file PICAX.EXE. This requires the presence of the necessary libraries (again see the link file PICAX). The Microsoft link command is:

   LINK @PICAX

   where PICAX is the input file to the linker.

4) Retain the file PICAX.EXE. All other files may be deleted. The program is started by typing PICAX (and then a return).

BATCH FILES

FORT.BAT

```
forl %1;
pas2
```

PICAX

```
picaxr+ittinr+user5+camv3+dtinpl+waitl+inp+out+
picax1+picax2+picax3+userr+user1+user2+user3+
user4,picax,,dos2for+fortran+grafsub+grafms2
```

The filenames before the first comma (on the third line) are the necessary object files and the filenames after the third comma are the needed library files (for example: DOS2FOR.LIB).

ATTACHMENT 3

example

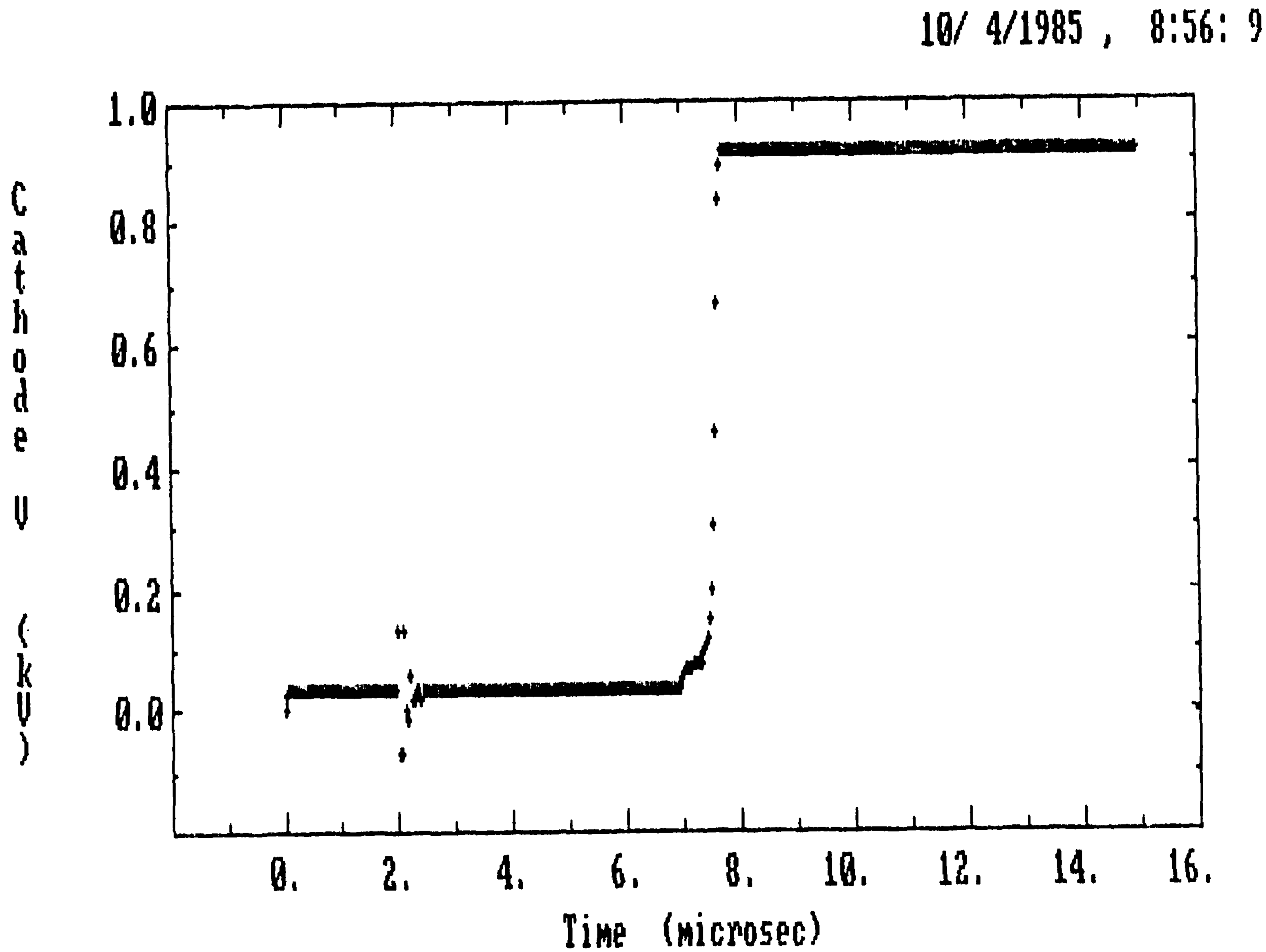
```
C>picax
PICAX     VERSION 2.00     12 September 1985
*help
 ERROR  8 - COMMANDS ARE ONE OR TWO LETTERS FOLLOWED BY
UP TO 4 FLOATING POINT NUMBERS (SEPARATED BY COMMAS).
FOR A LIST OF VALID COMMANDS TYPE LC.
*lc
 1.  a - Add to a user variable
 2.  an - Analyze data
 3.  cl - Close the disk output file
 4.  di - Directory listing
 5.  do - Do loop
 6.  e - Erase screen
 7.  ex - Exit program
 8.  fi - Find an old disk file for input
 9.  go - Go to program line
10.  he - Help the user out
11.  hc - Make hardcopy
12.  k - Kill program
13.  lc - List commnds
14.  lp - List program
15.  lv - List variables
16.  o - Open a new disk file for output
17.  pl - Plot data
18.  pa - Pause
19.  pr - Proceed with experiment
20.  r - Read a record from disk
21.  s - Start experiment
22.  t - Enter title
23.  v - Set a user variable
24.  w - Write a record to disk
25.  wa - Wait in units of 10 msec
26.  z - Zero or initialize data
27.  tr - Trigger on/off
*lv
 Title:
Experiment date and time :   0/ 0/   0 ,  0: 0: 0
List of experimental parameters.
VAR(  1) =       .00000      Initial cavity magnetic field (kG).
VAR(  2) =       .00000      Final cavity magnetic field (kG).
VAR(  3) =            0      Number of steps.
VAR(  4) =       .00000      Initial cavity field taper (%).
VAR(  5) =       .00000      Final cavity field taper (%).
VAR(  6) =            0      Number of steps.
VAR(  7) =       .00000      Initial e-beam voltage (kV).
VAR(  8) =       .00000      Final e-beam voltage (kV).
VAR(  9) =            0      Number of steps.
VAR( 10) =       .00000      Initial e-beam current (A).
VAR( 11) =       .00000      Final e-beam current (A).
VAR( 12) =            0      Number of steps.
VAR( 13) =       .00000      Initial e-beam alpha.
VAR( 14) =       .00000      Final e-beam alpha.
VAR( 15) =            0      Number of steps.
VAR( 16) =            0      Number of inteferometer points.
VAR( 17) =            0      Number of shots/pt. to avg. over.
VAR( 18) =            0      Number of pts. for tran. rec. 1.
```

```
VAR( 19) =              0       Number of pts. for tran. rec. 2.
VAR( 20) =              0       Number of pts. for tran. rec. 3.
VAR( 21) =              0       Number of pts. for tran. rec. 4.
VAR( 22) =              0       Number of pts. for tran. rec. 5.

*fi?
 FIND AN OLD DISK INPUT FILE.
EX:  "FI N M"
WILL CAUSE PROGRAM TO ASK FOR "FILENAME".
REPLY WITH ANY LEGAL NAME, SUCH AS "datal.dat".
THE FILE WILL BE OPENED TO LOGICAL UNIT NUMBER N,
WITH RECORD LENGTH M.  THE DEFAULT VALUES FOR N AND M
(IF SET TO 0) ARE LUN 3 AND RECORD LENGTH 11700 BYTES.
*fi
 Enter filename:  sl
File sl                          is open
*r
 Record number   1 is  11700 bytes long.
*lv
 Title: User variable setup file.
Experiment date and time :   9/27/1985 , 10: 8:55
List of experimental parameters.
VAR(  1) =       45.00000       Initial cavity magnetic field (kG).
VAR(  2) =       45.00000       Final cavity magnetic field (kG).
VAR(  3) =              1       Number of steps.
VAR(  4) =         .00000       Initial cavity field taper (%).
VAR(  5) =         .00000       Final cavity field taper (%).
VAR(  6) =              1       Number of steps.
VAR(  7) =         .00000       Initial e-beam voltage (kV).
VAR(  8) =         .00000       Final e-beam voltage (kV).
VAR(  9) =              0       Number of steps.
VAR( 10) =         .00000       Initial e-beam current (A).
VAR( 11) =         .00000       Final e-beam current (A).  .
VAR( 12) =              0       Number of steps.
VAR( 13) =         .00000       Initial e-beam alpha.
VAR( 14) =         .00000       Final e-beam alpha.
VAR( 15) =              0       Number of steps.
VAR( 16) =              0       Number of interferometer points.
VAR( 17) =              5       Number of shots/pt. to avg. over.
VAR( 18) =            450       Number of pts. for tran. rec. 1.
VAR( 19) =              0       Number of pts. for tran. rec. 2.
VAR( 20) =              0       Number of pts. for tran. rec. 3.
VAR( 21) =              0       Number of pts. for tran. rec. 4.
VAR( 22) =              0       Number of pts. for tran. rec. 5.
*T
 TITLE: example
* a 2 2
 * v 17 4
 *lv
 Title: example
Experiment date and time :   9/27/1985 , 10: 8:55
List of experimental parameters.
VAR(  1) =       45.00000       Initial cavity magnetic field (kG).
VAR(  2) =       47.00000       Final cavity magnetic field (kG).
VAR(  3) =              1       Number of steps.
VAR(  4) =         .00000       Initial cavity field taper (%).
VAR(  5) =         .00000       Final cavity field taper (%).
```

```
VAR(  6) =              1       Number of steps.
VAR(  7) =         .00000       Initial e-beam voltage (kV).
VAR(  8) =         .00000       Final e-beam voltage (kV).
VAR(  9) =              0       Number of steps.
VAR( 10) =         .00000       Initial e-beam current (A).
VAR( 11) =         .00000       Final e-beam current (A).
VAR( 12) =              0       Number of steps.
VAR( 13) =         .00000       Initial e-beam alpha.
VAR( 14) =         .00000       Final e-beam alpha.
VAR( 15) =              0       Number of steps.
VAR( 16) =              0       Number of interferometer points.
VAR( 17) =              4       Number of shots/pt. to avg. over.
VAR( 18) =            450       Number of pts. for tran. rec. 1.
VAR( 19) =              0       Number of pts. for tran. rec. 2.
VAR( 20) =              0       Number of pts. for tran. rec. 3.
VAR( 21) =              0       Number of pts. for tran. rec. 4.
VAR( 22) =              0       Number of pts. for tran. rec. 5.
*lv?
 LIST VARIABLES
LIST THE USER VARIABLES, THEIR VALUES, AND A BRIEF DESCRIPTION.
EX: "LV N"
WILL LIST THE FOLLOWING:
N           VARIABLES
0           EXPERIMENTAL PARAMETERS
1           A-D CHANNEL NUMBERS
2           TRANSIAC D-A CHANNEL NUMBERS
3           CAMAC SLOT NUMBERS
4           TRANSIENT RECORDER ATTENUATOR NUMBERS
5           PLOT VARIABLES
6           TRANSIENT RECORDER NUMBERS
7           DATA TRANSLATION CALIBRATION FACTORS
8           TRANSIENT RECORDER CALIBRATION FACTORS
10          TRANSIENT RECORDER ATTENUATOR VALUES
11          DATA TRANSLATION DATA
12          TRANSIENT RECORDER DATA
* 10 s
 * 20 e
 * 30 pl
 * 40 pa
 * 50 hc
 * 60 w
 * 55 e
 *lp
  10  s      .0000E+00     .0000E+00     .0000E+00     .0000E+00
   20  e    .0000E+00     .0000E+00     .0000E+00     .0000E+00
   30  pl   .0000E+00     .0000E+00     .0000E+00     .0000E+00
   40  pa   .0000E+00     .0000E+00     .0000E+00     .0000E+00
   50  hc   .0000E+00     .0000E+00     .0000E+00     .0000E+00
   55  e    .0000E+00     .0000E+00     .0000E+00     .0000E+00
   60  w    .0000E+00     .0000E+00     .0000E+00     .0000E+00
* 0
 Enter filename: junk.dat
File already exists
* 0
 Enter filename: junkl.dat
File junkl.dat            is open
```

```
* go
 * System not ready yet.
Enter 1 to bypass checks, 0 to continue checking.
1
<return>
```


10/ 4/1985 , 8:56: 9
Cathode V (kV)
1.0
0.8
0.6
0.4
0.2
0.0
0.
2.
4.
6.
8.
10.
12.
14.
16.
Time (microsec)

```
Record number   1 is   4334 bytes long
* an
 The microwave diode is unavailable from the transient recorder.
Enter the pulse width (sec) : 1e-6
Enter rep rate (Hz) : 100

Diode pulse height =    .000E+00 volts
Pulse width = 1.000E-06 seconds
Rep rate = 1.000E+02  Hz
Cathode voltage = -1.782E+00  kV
Collector current = -2.115E+00  Amps

Peak power = -1.864E+01  kW
Efficienty = -4.944E+02  %
* k
 * lp
 *^C ERROR 1 - OOPS.  YOU TYPED A CONTROL C.  TYPE EX TO EXIT.
* cl
 * ex
C>
```

ATTACHMENT 4

Table of Contents

```
c       PICAXR.FOR
C
c
$storage:2
c
c
      PROGRAM PICAX
C
C-----------------------------------------------------------------------
C
C       P I C A X    -    PROGRAM FOR INTERACTIVE CONTROL AND
C                          ACQUISITION FOR EXPERIMENTS
C
C-----------------------------------------------------------------------
C
C       ROOT MODULE
C
C-----------------------------------------------------------------------
C
C       PICAX IS A SET OF ROUTINES DESIGNED TO PROVIDE THE USER WITH
C       MANY OF THE OPERATIONS REQUIRED IN A GENERAL LABORATORY DATA
C       ACQUISITION AND ANALYSIS ENVIRONMENT.  PICAX PROVIDES A
C       HIGH-LEVEL INTERACTIVE COMMAND LANGUAGE WHICH CALLS
C       USER-WRITTEN SUBROUTINES WHICH IN TURN CALL ROUTINES IN
C       THE PICAX LIBRARY.  FOR MORE INFORMATION SEE THE PICAX
C       USER'S GUIDE.
C
C       WRITTEN BY ROBERT WALRAVEN
C       DEPARTMENT OF APPLIED SCIENCE
C       UNIVERSITY OF CALIFORNIA, DAVIS
C       LAST MODIFICATION ON 5 DEC 81
C
C-----------------------------------------------------------------------
C
c       This program was modified to run on an IBM PC-XT
c       by :
c       Tom Hargreaves
c       JAYCOR
c       205 S. Whiting St.
c       Alexandria, VA 22304
c
c       The original program is in capital letters,while
c       all modifications are in small letters.
c
c       Last modification :     September 26, 1985
c
c-----------------------------------------------------------------------
  c
        COMMON /P FLAGS/ PROG ON, EXPT ON, QUERY
        LOGICAL PROG ON, EXPT ON, QUERY
        COMMON /P INTER/ LINE, ICMND
```

```
      COMMON /P MATCH/ MCMND, NMATCH, CHARS, mmchar                     2
      common /p c var / n c var, cvar (4)
      logical input, interp
      character chars*2, icmnd*2
c
c      Initialize graphics and clear the screen.
C
      CALL qsmode ( 6 )
      call qcolor (11,0)
c
c      Initialize PICAX.
c
      CALL INIT
C
10    WRITE (*,20)
20    FORMAT(1X,'*'$)
30    IF ( INPUT(NUM CHR) ) GO TO 50
40    IF (EXPT ON) then
      CALL UPDATE
      go to 30
      endif
      IF (.NOT. PROG ON) GO TO 30
      CALL XECUTE(NMATCH)
      CALL NXT LN
      IF (PROG ON) GO TO 30
      GO TO 10
50    EXPT ON = .FALSE.
      PROG ON = .FALSE.
      IF (NUM CHR .EQ.  0) GO TO 10
      IF (NUM CHR .NE. -1) GO TO 60
      CALL ERROR(1)
      GO TO 10
60    IF (INTERP()) GO TO 70
      CALL ERROR(3)
      GO TO 10
70    IF (LINE.EQ.0 .OR. ICMND.NE.char(0)) GO TO 80
      CALL DELETE
      GO TO 10
80    NMATCH = 0
      mcmnd = 0
      CALL SEARCH
      IF (NMATCH .EQ. 0) GO TO 90
      NUM ERR = 0
      IF (LINE .EQ. 0) CALL XECUTE(NMATCH)
      IF (LINE .NE. 0) CALL INSERT
      GO TO 10
90    NUM ERR = NUM ERR + 1
      IF (NUM ERR .EQ. 3) GO TO 100
      CALL ERROR(2)
      GO TO 10
100   NUM ERR = 0
      CALL XECUTE (13)
      GO TO 10
      END
```

```
      SUBROUTINE XECUTE (N)
C
C---------------------------------------------------------------------
C
C       EXECUTES SYSTEM COMMANDS
C
C       THIS ROUTINE IS PART OF THE PICAX PROGRAM
C       WRITTEN BY ROBERT WALRAVEN, UCD - DAVIS
C       LAST MODIFICATION ON 19 SEP 81
C
C---------------------------------------------------------------------
C
      COMMON /P C VAR  / NCVAR, CVAR(4)
      COMMON /P DO VAR / DO FRST, DO LAST, DO CNT
      COMMON /U VAR    / N U VAR, U VAR (150)
      common/p files/nopen,lunopn(6),lunfnd(6),nfound,
     c                irecl(6),irnr(6),irnw(6),irnwmx(6)
      COMMON /P FLAGS  / PROG ON, EXPT ON, QUERY
      COMMON /P INTER  / LINE, ICMND
      COMMON /P MATCH  / MCMND, NMATCH, CHARS, mmchar
      COMMON /PROGRM   / NLINES, N LN MAX, PROG(50)
      COMMON /P VAR    / NCTRLC, LN PTR, P VERSN
      COMMON /P TITLE  / LTITLE (36)
      LOGICAL PROG ON, EXPT ON, QUERY
      logical lexist
      INTEGER DO FRST, DO LAST, DO CNT
      character key*1, ltitle*2, chars*2, icmnd*2, fname*20
C
      IF (.NOT.QUERY) GO TO 10
      CALL INFO(N)
      QUERY = .FALSE.
      RETURN
10    GO TO (100,200,300,400,500,600,700,800,900,1000,1100,1200,1300,
     1 1400,1500,1600,1700,1800,1900,2000,2100,2200,2300,2400,2500,
     2 2600,2700,2800),N
C
C------------------------------------ADD TO A GENERAL VARIABLE
100   I = CVAR(1)
      IF (I.GE.1 .AND. I.LE.NUVAR) GO TO 130
      CALL ERROR(4)
      RETURN
130   UVAR(I) = UVAR(I) +CVAR(2)
      RETURN
C------------------------------------ANALYZE DATA
200   CALL UANLYZ
      RETURN
C------------------------------------CLOSE THE DISK OUTPUT FILE
300   LUN = 2
      IF (CVAR(1) .NE. 0.) LUN = CVAR(1)
      IF (NOPEN .NE. 0) GO TO 305
      CALL ERROR (16)
      return
305   DO 310 I=1,NOPEN
      IF (LUN .EQ. LUNOPN(I)) GO TO 320
```

```
310   CONTINUE
      CALL ERROR (17)
      RETURN
320   write(lun,err=360,rec=(irnwmx(i)+1))lun
      backspace lun
      endfile lun
      CLOSE(LUN,iostat=ioerr)
      if(ioerr) 340,330,350
330   IF (I .EQ. NOPEN) LUNOPN (NOPEN) = 0
      LUNOPN(I) = LUNOPN(NOPEN)
      NOPEN = NOPEN - 1
      return
340   write(*,'('' End of file on close operation.'')')
      return
350   write(*,'('' Error on close operation.'')')
      return
360   write(*,'('' Error on write.'')')
      RETURN
C-----------------------------------DIRECTORY LISTING
400   I = CVAR(1)
      IF (I.NE.6) I=5
      CALL qsmode ( 6 )
      call qcolor (11,0)
      write (*,410)
410   format(' Directory listing not implemented')
      RETURN
C-----------------------------------DO LOOP
500   I = CVAR(1)
      DO LAST = LINE NM (I)
      IF (DO LAST .NE. 0) GO TO 520
      CALL ERROR(5)
      RETURN
520   DO CNT = CVAR(2)
      DO FRST = LN PTR
      RETURN
C-----------------------------------ERASE SCREEN
600   CALL qsmode ( 6 )
      call qcolor (11,0)
      RETURN
C-----------------------------------EXIT
700   IF (NOPEN .EQ. 0) GOTO 720
      DO 710, I=1,NOPEN
      write(lunopn(i),err=715,rec=(irnwmx(i)+1)) lun
      backspace lunopn(i)
      endfile lunopn(i)
      CLOSE(LUNOPN(I),iostat=ioerr)
710   if(ioerr .ne. 0)write(*,'('' Error closing logical unit ''i3)')
     c       lunopn(i)
      go to 720
715   write(*,'('' Error writing to logical unit ''i3)')lunopn(i)
720   continue
      call qsmode (3)
      call qclear (0,7)
      continue
```

```
      CALL EXIT
C-------------------------------------FIND OLD DISK INPUT FILE
800   lun = 3
      ioerr = 0
      if (cvar(1) .ne. 0.0) lun = cvar(1)
      if (nfound .eq. 0) go to 830
      do 810 i=1,nfound
      if (lun .eq. lunfnd(i)) go to 820
810   continue
      go to 830
820   close (lun,iostat=ioerr)
      if(ioerr) 860,825,870
825   if (i .eq. nfound) lunfnd(nfound) = 0
      lunfnd(i) = lunfnd(nfound)
      nfound = nfound - 1
830   if (nfound .eq. 6) go to 840
      write (*,'('' Enter filename:  '',$)')
      read (*,'(a20)') fname
      inquire (file=fname, exist=lexist)
      if (.not. lexist) then
      call error (15)
      return
      endif
      irecl(nfound+1) = 11700
      if(cvar(2) .ne. 0.0)irecl(nfound+1) = cvar(2)
      open(lun,file=fname,status='old',access='direct',iostat=ioerr,
     *       recl=irecl(nfound+1))
      if (ioerr .eq. 0) then
      read(lun,iostat=ioerr,err=850,rec=1)irecln
      rewind lun
      if(irecln .ne. irecl(nfound+1)) then
      write(*,'('' Incorrect record length, the correct length is ''
     c       i6)') irecln
      close (lun)
      return
      endif
      nfound = nfound + 1
      lunfnd(nfound) = lun
      irnr (nfound) = 1
      write (*,'('' File '',a20,''  is open'')') fname
      return
      endif
840   call error (6)
      if (ioerr .lt. 0) write (*,'('' End of file'')')
      return
850   write(*,'('' Error on test read.'')')
      return
860   write(*,'('' End of file on close operation.'')')
      return
870   write(*,'('' Error on close operation.'')')
      RETURN
C-------------------------------------GO
900   I = CVAR(1)
      IF (I .EQ. 0) GO TO 910
```

```
      LN PTR = LINE NM(I)
      IF (LN PTR .NE. 0) GO TO 920
      CALL ERROR(7)
      RETURN
910   LN PTR = 1
920   PROG ON = .TRUE.
      RETURN
C------------------------------------------HELP
1000  CALL ERROR(8)
      RETURN
C------------------------------------------HARD COPY
1100  CALL pscrn
      RETURN
C------------------------------------------KILL PROGRAM
1200  NLINES = 0
      PROG ON = .FALSE.
      RETURN
C------------------------------------------LIST COMMANDS
1300  MCMND = -1
      CALL SEARCH
      MCMND = 0
      RETURN
C------------------------------------------LIST PROGRAM
1400  CALL PLIST
      RETURN
C------------------------------------------LIST VARIABLES
1500  CALL UVLIST
      RETURN
C------------------------------------------OPEN NEW DISK OUTPUT FILE
1600  if (nopen .eq. 6) go to 1640
      ioerr = 0
      lun = 2
      if (cvar(1) .ne. 0.0) lun = cvar(1)
      write (*,'('' Enter filename: '',$)')
      read (*,'(a20)') fname
      inquire (file = fname, exist = lexist)
      if (lexist) then
      write (*,'('' File already exists'')')
      return
      endif
      if (nopen .eq. 0) go to 1630
      do 1610 i=1,nopen
      if (lun .eq. lunopn(i)) go to 1620
1610  continue
      go to 1630
1620  close (lun,iostat=ioerr)
      if(ioerr) 1650,1625,1660
1625  if (i .eq. nopen) lunopn(nopen) = 0
      lunopn(i) = lunopn(nopen)
      nopen = nopen - 1
1630  irecl (nopen+1) = 11700
      if(cvar(2) .ne. 0.0)irecl(nopen+1) = cvar(2)
      open(lun,file=fname,status='new',access='direct',iostat=ioerr,
     *       recl=irecl(nopen+1))
```

```
      if (ioerr .ne. 0) then
      call error (9)
      if (ioerr .lt. 0)write(*,'('' End of file'')')
      return
      endif
      write (lun, err=1640, rec=1) lun
      rewind (lun)
      nopen = nopen + 1
      lunopn (nopen) = lun
      irnw (nopen) = 1
      irnwmx (nopen) = 0
      write(*,'('' File '',a20,'' is open'')') fname
      return
1640  call error(9)
      return
1650  write(*,'('' End of file on close operation.'')')
      return
1660  write(*,'('' Error on close operation.'')')
      RETURN
C-------------------------------------PLOT DATA
1700  CALL UPLOT
      RETURN
C-------------------------------------PAUSE
1800  call inkey ( key )
      num = ichar(key)
      IF ( num .NE. 13 ) GO TO 1800
      RETURN
C-------------------------------------PROCEED
1900  EXPT ON = .TRUE.
      CALL UPROCD
      RETURN
C-------------------------------------READ
2000  LUN = 3
      IF (CVAR(1) .NE. 0.0) LUN = CVAR(1)
      IF (NFOUND .EQ. 0) GO TO 2020
      DO 2010, I=1,NFOUND
      IF (LUN .EQ. LUNFND(I)) GO TO 2030
2010  CONTINUE
2020  CALL ERROR(15)
      RETURN
2030  CALL UREAD(LUN,1)
      RETURN
C-------------------------------------START
2100  EXPT ON = .TRUE.
      CALL USTART
      RETURN
C-------------------------------------TITLE
2200  CALL TITLE
      RETURN
C-------------------------------------SET A VARIABLE
2300  I = CVAR(1)
      IF (I.GE.1 .AND. I.LE.NUVAR) GO TO 2310
      CALL ERROR(4)
      RETURN
2310  UVAR(I) = CVAR(2)
      RETURN
C-------------------------------------WRITE
```

```
2400  LUN = 2
      IF (CVAR(1) .NE. 0.0) LUN = CVAR(1)
      IF (NOPEN .EQ. 0) GO TO 2420
      DO 2410, I=1,NOPEN
      IF (LUN .EQ. LUNOPN(I)) GO TO 2430
2410  CONTINUE
2420  CALL ERROR(13)
      RETURN
2430  CALL UWRITE(LUN,1)
      RETURN
C----------------------------------------WAIT
2500  I = CVAR(1)
      CALL WAIT (I)
      RETURN
C----------------------------------------ZERO
2600  CALL UZERO
      RETURN
C------------------------------------USER COMMANDS
2700  CALL UCMNDS (N)
      RETURN
C
2800  RETURN
C
      END
      SUBROUTINE NXT LN
C
C-----------------------------------------------------------------
C
C      GETS NEXT LINE OF PROGRAM
C
C      THIS ROUTINE IS PART OF THE PICAX PROGRAM
C      WRITTEN BY ROBERT WALRAVEN, UCD - DAVIS
C      LAST MODIFICATION ON 29 JUN 81
C
C-----------------------------------------------------------------
C
      COMMON /P C VAR  / NCVAR, CVAR(4)
      COMMON /P DO VAR / DO FRST, DO LAST, DO CNT
      COMMON /P FLAGS  / PROG ON, EXPT ON, QUERY
      COMMON /P MATCH  / MCMND, NMATCH, CHARS, mmchar
      COMMON /PROGRM   / NLINES, N LN MAX, PROG(50)
      COMMON /P VAR    / NCTRLC, LN PTR, P VERSN
      LOGICAL PROG ON, EXPT ON, QUERY
      INTEGER DO FRST, DO LAST, DO CNT
      character chars*2
      DIMENSION N(2)
      EQUIVALENCE (PACK,N(1))
C
      IF (.NOT. PROG ON) RETURN
      IF (DO CNT .EQ. 0) GO TO 10
      IF (LN PTR .LE. DO LAST) GO TO 10
      DO CNT = DO CNT - 1
      IF (DO CNT .NE. 0) LN PTR = DO FRST
10    IF (LN PTR .LT. NLINES+1) GO TO 20
```

```
      PROG ON = .FALSE.
      RETURN
20    INDEX = LNPTR*5 - 4
      PACK = PROG (INDEX)
      NMATCH = N(2)
      DO 30 I=1,4
30    CVAR(I) = PROG (INDEX + I)
      LN PTR = LN PTR + 1
      RETURN
      END
      LOGICAL FUNCTION INPUT(N)
C
C-----------------------------------------------------------------------
C
C       INPUTS A STRING OF TEXT OF UP TO 72 CHARACTERS FROM CONSOLE.
C
C       WHILE TEXT IS BEING TYPED, THE SYSTEM CONSOLE HANDLER HOLDS
C       THE INPUT.  WHEN A RETURN OR CONTROL-C IS TYPED, THE TEXT IS
C       THEN AVAILABLE TO THE USER.  THIS ROUTINE GETS A CHARACTER
C       AT A TIME FROM THE HANDLER BY CALLING THE SYSTEM ROUTINE
C       ITTINR.  ON RETURN, IF TEXT IS AVAILABLE, IT IS PUT IN THE
C       VARIABLE 'STRING' AND 'NCHAR' IS THE NUMBER OF CHARACTERS
C       TYPED (NOT COUNTING RETURN).  IF A CONTROL-C WAS TYPED, NCHAR
C       WILL HAVE A VALUE OF -1.
C
C       INPUT = .FALSE. IF NO TEXT IS AVAILABLE
C             = .TRUE.  IF TEXT RETURNED
C
C       THIS ROUTINE IS PART OF THE PICAX PROGRAM
C       WRITTEN BY ROBERT VALRAVEN, UCD - APPLIED SCIENCE
C       LAST MODIFICATION ON  4 AUG 80
C
C-----------------------------------------------------------------------
C
      COMMON /P INPUT  / NCHAR, STRING(72), NPTR
      character*1 STRING, I
C
      INPUT = .FALSE.
      j = ittinr ()
      IF (j .EQ. 0) RETURN
      INPUT = .TRUE.
      NPTR = 1
      NCHAR = 1
10    IF (j.EQ.13 .OR. j.EQ.10) GO TO 20
      IF (j .EQ. 3) GO TO 30
      if(j.ge.65 .and. j.le.90)j = j + 32
      i = char (j)
      STRING(NCHAR) = I
      if (j .eq. 8) nchar = nchar - 2
      if (nchar .lt. 0) nchar = 0
      IF (NCHAR .EQ. 71) GO TO 40
      NCHAR = NCHAR + 1
15    j = ittinr ()
      if (j .eq. 0) go to 15
```

```
      GO TO 10
20    NCHAR = NCHAR - 1
      write (*,'(/$)')
      GO TO 40
30    NCHAR = -1
40    N = NCHAR
      STRING(NCHAR+1) = 0
      RETURN
      END
C       PICAX1.FOR
C
c
$storage:2
c
c
C-----------------------------------------------------------------
C
C       PICAX OVERLAY MODULE - REGION 1
C
C-----------------------------------------------------------------
C
      LOGICAL FUNCTION INTERP()
C
C-----------------------------------------------------------------
C
C       INTERPRETS A LINE TYPED BY THE USER.  THE FORM OF THE LINE
C       MUST BE
C       [LINE NUMBER] COMMAND [NUMBER[,NUMBER[,NUMBER[,NUMBER]]]]
C       IF A VALID LINE IS TYPED, ON RETURN INTERP IS TRUE,
C       OTHERWISE IT IS FALSE.
C
C       THIS ROUTINE IS PART OF THE PICAX PROGRAM
C       WRITTEN BY ROBERT WALRAVEN, UCD - APPLIED SCIENCE
C       LAST MODIFICATION ON 5 DEC 81
C
C-----------------------------------------------------------------
C
c       This routine was modified to run on an IBM PC-XT
c       by :
c       Tom Hargreaves
c       JAYCOR
c       205 S. Whiting St.
c       Alexandria, VA 22304
c       Last modification on 5 Oct 84
c
c----------------------------------------------------------------
c
      COMMON /P FLAGS / PROG ON, EXPT ON, QUERY
      COMMON /P INTER / LINE, ICMND
      COMMON /P C VAR / NCVAR, CVAR(4)
      LOGICAL PROG ON, EXPT ON, QUERY
      logical number, ncnvrt
      character IN(2)*1, i*1, icmnd*2
      EQUIVALENCE (IN(1),ICMND)
```

```
C
C       LOOK FOR LINE NUMBER
C
      LINE = 0
      ICMND = char(0)
      INTERP = .TRUE.
      DO 1 II=1,NCVAR
 1    CVAR(II) = 0.
10    II = NEXT(0)
      I = char(II)
      IF (II.LT.0) RETURN
      IF (.NOT.NUMBER(Ii)) GO TO 30
      LINE = LINE*10 + II
      IF (LINE.LE.999) GO TO 10
      CALL ERROR(10)
      RETURN
C
C       GET COMMAND
C
30    IF (I .NE. ' ') GO TO 35
      ii = next (0)
      I = char (ii)
      GO TO 30
35    IN(1) = I
      IN(2) = ' '
      II = NEXT(-1)
      I = char( II )
      IF (II.LT.0) RETURN
      IF (I.LT.'a' .OR. I.GT.'z') GO TO 40
      IN(2) = I
C
C
C       SKIP REST OF WORD
C
36    II = NEXT(0)
      II = NEXT(-1)
      I = char( II )
      IF (I.GE.'a' .AND. I.LE.'z') GO TO 36
C
C       GET VARIABLES
C
40    IF (I .EQ. '?') QUERY = .TRUE.
      IF (QUERY) RETURN
      IF (II.LT.0) RETURN
45    DO 50 II=1,NCVAR
      IF (.NOT.NCNVRT(X)) INTERP = .FALSE.
50    CVAR(II) = X
      RETURN
      END
c
      LOGICAL FUNCTION MATCH (NCMND, nstng, CSTRNG)
C
C--------------------------------------------------------------------
C
```

```
C      IF MCMND = 0, COMPARES ICMND WITH NSTRNG.
C                    IF ICMND =  NSTRNG, THEN NMATCH = NCMND,
C                                        AND  MATCH  = .TRUE.
C                    IF ICMND <> NSTRNG, MATCH = .FALSE.
C      IF MCMND = -1, PRINTS NCMND, MSTRNG, AND CSTRNG ON CONSOLE.
C      IF MCMND =  1, COMPARES nmatch WITH NCMND
C
C      THIS ROUTINE IS PART OF THE PICAX PROGRAM
C      WRITTEN BY ROBERT WALRAVEN, UCD - APPLIED SCIENCE
C      LAST MODIFICATION ON  4 AUG 80
C
C-----------------------------------------------------------------------
C
      COMMON /P MATCH / MCMND, NMATCH, CHARS, mmchar
      COMMON /P INTER / LINE, ICMND
      character*1 NSTRNG(4), OUTPUT(72), MCHAR(4)
      character CSTRNG*35, icmnd*2, mstrng*2, chars*2, nstng*4,nsng*4
      EQUIVALENCE (MSTRNG,MCHAR(1))
      equivalence (nsng,nstrng(1))
      EQUIVALENCE (CHARS ,MCHAR(1))
C
      MATCH = .FALSE.
      nsng = nstng
      DO 1 I=1,4
1     MCHAR(I) = NSTRNG(I)
      IF (MCMND .EQ.  0) GO TO  10
      IF (MCMND .EQ.  1) GO TO  20
      IF (MCMND .EQ. -1) GO TO 100
      RETURN
10    IF (ICMND .NE. MSTRNG) RETURN
      NMATCH = NCMND
      MATCH = .TRUE.
      RETURN
20    IF (nmatch .EQ. NCMND) MATCH = .TRUE.
      RETURN
100   IF (NCMND .EQ.  1) CALL qsmode ( 6 )
      if (ncmnd .eq. 1) call qcolor (11,0)
      WRITE (*,110) NCMND, MSTRNG
110   FORMAT(1X,I2'.',2X,A2' - '$)
      write (*,120) cstrng
120   format (1x,a35)
      RETURN
      END
c
      LOGICAL FUNCTION NCNVRT(X)
C
C-----------------------------------------------------------------------
C
C      CONVERTS A FREE FIELD ASCII STRING INTO A NUMBER.
C      THE ASCII STRING IS SCANNED UNTIL AN ILLEGAL
C      CHARACTER, A CARRIAGE RETURN, OR A COMMA IS ENCOUNTERED.
C      THE NUMBER IS RETURNED IN X.  IF THE STRING IS AN
C      INVALID NUMBER, NCNVRT IS RETURNED .FALSE.
C
```

```
C      THIS ROUTINE IS PART OF THE PICAX PROGRAM                    13
C      WRITTEN BY ROBERT WALRAVEN, UCD - APPLIED SCIENCE
C      LAST MODIFICATION ON  4 AUG 80
C
C-----------------------------------------------------------------------
C
      logical decflg, number
      character*1 ICHARr
C
      NCNVRT = .TRUE.
      NCOMMA = 0
      NDEC = 0
      DECFLG = .FALSE.
      SIGN = 1.0
      NESIGN = 1
      X = 0.
1     i = next (0)
      ICHARr = char( i )
      IF (i .LT. 0) RETURN
      IF (ICHARr .EQ. ' ') GO TO 1
      IF (ICHARr .EQ. '+') GO TO 10
      IF (ICHARr .NE. '-') GO TO 20
      SIGN = -1.
10    i = next (0)
      ICHARr = char( i )
      IF (i .LT. 0) GO TO 1000
20    IF (.NOT.NUMBER( i )) GO TO 50
      ichar1 = i
      X = X*10. + ICHAR1
      IF (DECFLG) NDEC = NDEC+1
30    i = next (0)
      ICHARr = char( i )
      IF (ICHARr.EQ.',' .OR. ICHARr.EQ.' ') GO TO 40
      IF (ICHARr .EQ. '=') GO TO 40
      IF (i .GE. 0   ) GO TO 20
40    X = SIGN*X/(10.**NDEC)
      RETURN
50    IF (ICHARr .NE. '.') GO TO 60
      IF (DECFLG) GO TO 1000
      DECFLG = .TRUE.
      GO TO 30
60    IF (ICHARr .EQ. ',' .or. icharr .eq. ' ') GO TO 40
      IF (ICHARr .NE. 'e') GO TO 1000
      i = next (0)
      ICHARr = char( i )
      IF (i .LT. 0) GO TO 1000
      IF (ICHARr .EQ. '+') GO TO 70
      IF (ICHARr .NE. '-') GO TO 80
      NESIGN = -1
70    i = next (0)
      ICHARr = char( i )
      IF (i .LT. 0) GO TO 1000
80    IF (.NOT.NUMBER( i )) GO TO 1000
      NEXP = i
```

```
      i = next (0)
      ICHARr = char( i )
      IF (ICHARr .EQ. ',' .or. icharr .eq. ' ') GO TO 90
      IF (i .GE. 0) GO TO 100
90    X = SIGN*X/(10.**(NDEC-NESIGN*NEXP))
      RETURN
100   IF (.NOT.NUMBER( i )) GO TO 1000
      ichar1 = i
      NEXP = NEXP*10 + ICHAR1
      i = next (0)
      ICHARr = char( i )
      IF (NEXP .LE. 20) GO TO 90
1000  NCNVRT = .FALSE.
      X = 0.
      RETURN
      END
c
      FUNCTION NEXT(I)
C
C-----------------------------------------------------------------
C
C      GETS THE NEXT CHARACTER FROM THE TEXT BUFFER.  ON EXIT, 'NEXT'
C      CONTAINS THE CHARACTER AND THE TEXT BUFFER POINTER 'NPTR' IS
C      UPDATED BY 1+I CHARACTERS.  IF NO CHARACTER IS AVAILABLE,
C      'NEXT' IS SET EQUAL TO -1.
C
C      THIS ROUTINE IS PART OF THE PICAX PROGRAM
C      WRITTEN BY ROBERT WALRAVEN, UCD - APPLIED SCIENCE
C      LAST MODIFICATION ON  4 AUG 80
C
C-----------------------------------------------------------------
C
c
c      Modified so that 'NEXT' contains the ascii code
c      of the character.
c
c      Last modification on 17 Oct 84
c
c-----------------------------------------------------------------
c
      COMMON /P INPUT / NCHAR, STRING(72), NPTR
      character*1 STRING
C
      IF (NPTR .LE. NCHAR) GO TO 10
      NEXT = -1
      RETURN
10    NEXT = ichar( STRING(NPTR) )
      NPTR = NPTR+1+I
      RETURN
      END
c
      LOGICAL FUNCTION NUMBER(I)
C
C-----------------------------------------------------------------
```

```
C
C      TEST THE VALUE OF I TO SEE IF IT IS A NUMERIC CHARACTER.
C      ON RETURN, IF
C            NUMERIC     : 'NUMBER' TRUE AND I=NUMERIC VALUE OF CHAR
C            NON-NUMERIC : 'NUMBER' FALSE AND I UNCHANGED
C
C      THIS ROUTINE IS PART OF THE PICAX PROGRAM
C      WRITTEN BY ROBERT WALRAVEN, UCD - APPLIED SCIENCE
C      LAST MODIFICATION ON  4 AUG 80
C
C-----------------------------------------------------------------------
C
      NUMBER = .FALSE.
      IF (I.LT.ichar( '0' ) .OR. I.GT.ichar( '9' )) RETURN
      NUMBER = .TRUE.
      I = I-48
      RETURN
      END
c
      SUBROUTINE SEARCH
C
C-----------------------------------------------------------------------
C
C      DEFINES VALID PICAX SYSTEM COMMANDS.  THE FIRST ARGUMENT
C      OF MATCH IS THE COMMAND NUMBER, THE SECOND ARGUMENT IS THE
C      ONE OR TWO LETTER COMMAND, AND THE THIRD ARGUMENT IS A
C      BRIEF DESCRIPTION OF THE COMMAND.
C
C      THIS ROUTINE IS PART OF THE PICAX PROGRAM
C      WRITTEN BY ROBERT WALRAVEN, UCD - APPLIED SCIENCE
C      LAST MODIFICATION ON 29 JUN 81
C
C-----------------------------------------------------------------------
C
      logical MATCH
c
      IF(match(1,'a   ','Add to a user variable                  '))RETURN
      IF(MATCH(2,'an  ','Analyze data                            '))RETURN
      IF(MATCH(3,'cl  ','Close the disk output file              '))RETURN
      IF(MATCH(4,'di  ','Directory listing                       '))RETURN
      IF(MATCH(5,'do  ','Do loop                                 '))RETURN
      IF(MATCH(6,'e   ','Erase screen                            '))RETURN
      IF(MATCH(7,'ex  ','Exit program                            '))RETURN
      IF(MATCH(8,'fi  ','Find an old disk file for input         '))RETURN
      IF(MATCH(9,'go  ','Go to program line                      '))RETURN
      IF(MATCH(10,'he  ','Help the user out                       '))RETURN
      IF(MATCH(11,'hc  ','Make hardcopy                           '))RETURN
      IF(MATCH(12,'k   ','Kill program                            '))RETURN
      IF(MATCH(13,'lc  ','List commands                           '))RETURN
      IF(MATCH(14,'lp  ','List program                            '))RETURN
      IF(MATCH(15,'lv  ','List variables                          '))RETURN
      IF(MATCH(16,'o   ','Open a new disk file for output         '))RETURN
      IF(MATCH(17,'pl  ','Plot data                               '))RETURN
      IF(MATCH(18,'pa  ','Pause                                   '))RETURN
```

```
      IF(MATCH(19,'pr  ','Proceed with experiment           '))RETURN
      IF(MATCH(20,'r   ','Read a record from disk           '))RETURN
      IF(MATCH(21,'s   ','Start experiment                  '))RETURN
      IF(MATCH(22,'t   ','Enter title                       '))RETURN
      IF(MATCH(23,'v   ','Set a user variable               '))RETURN
      IF(MATCH(24,'w   ','Write a record to disk            '))RETURN
      IF(MATCH(25,'wa  ','Wait in units of 10 msec          '))RETURN
      IF(MATCH(26,'z   ','Zero or initialize data           '))RETURN
      CALL UMATCH
      RETURN
      END
c
      FUNCTION LINE NM (LINE)
C
C-----------------------------------------------------------------
C
C      RETURN THE LINE COUNT FOR A LINE NUMBER 'LINE'.  IF THE
C      LINE DOES NOT EXIST, LINE NM IS RETURNED ZERO.
C
C      THIS ROUTINE IS PART OF THE PICAX PROGRAM
C      WRITTEN BY ROBERT WALRAVEN, UCD - APPLIED SCIENCE
C      LAST MODIFICATION ON  4 AUG 80
C
C-----------------------------------------------------------------
C
      COMMON /PROGRM  / NLINES, N LN MAX, PROG(50)
      EQUIVALENCE (PACK,N)
C
      LINE NM = 0
      IF (NLINES .NE. 0) GO TO 30
10    CALL ERROR(11)
      RETURN
30    DO 40 I=1,NLINES
      PACK = PROG(I*5-4)
      IF (N .EQ. LINE) GO TO 50
40    CONTINUE
      GO TO 10
50    LINE NM = I
      RETURN
      END
c
      subroutine wait(i)
c
c     This routine waits i increments of approximately
c     9.2 milliseconds before returning by printing nul
c     characters to the screen.
c
      character ichar*1
c
      ichar = char (0)
      do 10 j=1,i
10    write (*,'(a1,t11,$)') ichar
      return
      END
```

```
C       PICAX2.FOR
C
c
$storage:2
c
c
C-----------------------------------------------------------------------
C
C       PICAX OVERLAY MODULE - REGION 2
C
C-----------------------------------------------------------------------
      SUBROUTINE DEFINE (N, TYPE, STRING)
C
C-----------------------------------------------------------------------
C
C       PRINTS N, UVAR(N) AND DESCRIPTIVE ASCII STRING ON CONSOLE
C
C       THIS ROUTINE IS PART OF THE PICAX PROGRAM
C       WRITTEN BY ROBERT WALRAVEN, UCD - APPLIED SCIENCE
C       LAST MODIFICATION ON 17 JUL 81
C
C-----------------------------------------------------------------------
C
c       This program was modified for use with an IBM-PCXT
c       by :
c       Tom Hargreaves
c       JAYCOR
c       205 S. Whiting St.
c       Alexandria, VA 22304
c
c       Last modified :         September 26, 1985
c
c----------------------------------------------------------------------
c
      COMMON /U VAR   / N U VAR, U VAR (150)
      COMMON /P TITLE  / LTITLE (36)
      character TYPE*1, string*35, ltitle*2
      integer n
C
      IF (N .GT. NUVAR) RETURN
      VALUE = UVAR(N)
      WRITE (*,10) N
10    FORMAT(' VAR('I3') = ',$)
      IF (TYPE .EQ. 'i') IVALUE = VALUE
      IF (TYPE .EQ. 'i') WRITE (*,20) IVALUE
      IF (TYPE .EQ. 'r') WRITE (*,30) VALUE
      IF (TYPE .EQ. 'e') WRITE (*,40) VALUE
20    FORMAT(I12,5X$)
30    FORMAT(F12.5,5X$)
40    FORMAT(1PE12.5,5X$)
      write (*,50) string
50    format (1x,a35)
      RETURN
      END
```

```
      SUBROUTINE DELETE
C
C-----------------------------------------------------------------------
C
C       DELETE A LINE FROM THE PROGRAM
C
C       THIS ROUTINE IS PART OF THE PICAX PROGRAM
C       WRITTEN BY ROBERT WALRAVEN, UCD - APPLIED SCIENCE
C       LAST MODIFICATION ON  4 AUG 80
C
C-----------------------------------------------------------------------
C
      COMMON /P INTER  / LINE, ICMND
      COMMON /PROGRM   / NLINES, N LN MAX, PROG(50)
      character icmnd*2
C
      I = LINE NM(LINE)
      IF (I.EQ.0) RETURN
      IF (I .EQ. NLINES) GO TO 20
      DO 10 J=I,(NLINES-1)
      N = (J-1)*5
      DO 10 JJ=1,5
      NN = N+JJ
10    PROG (NN) = PROG (NN+5)
20    NLINES = NLINES - 1
      RETURN
      END
      SUBROUTINE ERROR (I)
C
C-----------------------------------------------------------------------
C
C       PRINT ERROR MESSAGES
C
C       THIS ROUTINE IS PART OF THE PICAX PROGRAM
C       WRITTEN BY ROBERT WALRAVEN, UCD - APPLIED SCIENCE
C       LAST MODIFICATION ON  19 SEP 81
C
C-----------------------------------------------------------------------
C
      COMMON /P FLAGS  / PROG ON, EXPT ON, QUERY
      LOGICAL PROG ON, EXPT ON, QUERY
      character ibell*1
      ibell = char (7)
C
      PROG ON = .FALSE.
      write (*,2) ibell
2     format (1x,a,$)
      WRITE (*,3) I
3     FORMAT(' ERROR 'I2' -'$)
      GO TO (10,20,30,40,50,60,70,80,90,100,110,120,130,140,150,
     1      160,170),I
C
10    WRITE (*,12)
12    FORMAT(' OOPS. YOU TYPED A CONTROL-C. TYPE EX TO EXIT.')
```

```
      RETURN
20    WRITE (*,22)
22    FORMAT(' THAT IS NOT A COMMAND. TYPE HE IF YOU NEED HELP.')
      RETURN
30    WRITE (*,32)
32    FORMAT(' MISTEAK-IN LINE.  PLEASE RETYPE IT.')
      RETURN
40    WRITE (*,42)
42    FORMAT(' VARIABLE INDEX OUT OF RANGE')
      RETURN
50    WRITE (*,52)
52    FORMAT(' DO LOOP TO NON-EXISTENT LINE')
      RETURN
60    WRITE (*,62)
62    FORMAT(' FIND ERROR')
      RETURN
70    WRITE (*,72)
72    FORMAT(' GO TO NON EXISTENT LINE')
      RETURN
80    WRITE (*,82)
82    FORMAT(' COMMANDS ARE ONE OR TWO LETTERS FOLLOWED BY'
     1      /' UP TO 4 FLOATING POINT NUMBERS (SEPARATED BY COMMAS).'
     2      /' FOR A LIST OF VALID COMMANDS TYPE LC.')
      RETURN
90    WRITE (*,92)
92    FORMAT(' OPEN ERROR')
      RETURN
100   WRITE (*,102)
102   FORMAT(' LINE NUMBER MUST BE BETWEEN 1 AND 999')
      RETURN
110   WRITE (*,112)
112   FORMAT(' NO SUCH LINE NUMBER')
      RETURN
120   WRITE (*,122)
122   FORMAT(' PROGRAM BUFFER FULL')
      RETURN
130   WRITE (*,132)
132   FORMAT(' OUTPUT FILE NOT OPEN')
      RETURN
140   WRITE (*,142)
142   FORMAT(' FILE ALREADY OPEN')
      RETURN
150   WRITE (*,152)
152   FORMAT(' NO INPUT FILE')
      RETURN
160   WRITE (*,162)
162   FORMAT(' NO FILES OPEN')
      RETURN
170   WRITE (*,172)
172   FORMAT(' NO FILE OPEN ON THAT LOGICAL UNIT')
      RETURN
C
      END
      SUBROUTINE INIT
```

```
C                                                                        20
C------------------------------------------------------------------
C
C      INITIALIZE PICAX VARIABLES
C
C      THIS ROUTINE IS PART OF THE PICAX PROGRAM
C      WRITTEN BY ROBERT WALRAVEN, UCD - APPLIED SCIENCE
C      LAST MODIFICATION ON 19 SEP 81
C
C------------------------------------------------------------------
C
      COMMON /P C VAR   / NCVAR, CVAR(4)
      COMMON /P DO VAR / DO FRST, DO LAST, DO CNT
      common/p files/nopen,lunopn(6),lunfnd(6),nfound,
     c                irecl(6),irnr(6),irnw(6),irnwmx(6)
      COMMON /P FLAGS / PROG ON, EXPT ON, QUERY
      COMMON /PROGRM   / NLINES, N LN MAX, PROG(50)
      COMMON /P TITLE  / LTITLE(36)
      COMMON /P VAR    / NCTRLC, LN PTR, P VERSN
      COMMON /U VAR    / N U VAR, U VAR (150)
      LOGICAL PROG ON, EXPT ON, QUERY
      INTEGER DO FRST, DO LAST, DO CNT
      character ltitle*2
C
      P VERSN = 2.0
      NCVAR = 4
      N LN MAX = 10
      NCTRLC = 0
      NLINES = 0
      DO CNT = 0
      PROG ON = .FALSE.
      NOPEN = 0
      NFOUND = 0
      DO 1 I=1,6
      LUN OPN (I) = 0
1     LUN FND (I) = 0
      EXPT ON = .FALSE.
      QUERY = .FALSE.
      DO 20 I=1,35
20    LTITLE(I) = '  '
      LTITLE(36) = char( 0 )
      WRITE (*,30) P VERSN
30    FORMAT(' PICAX     VERSION 'F5.2'     12 September 1985')
      CALL UINIT
      RETURN
      END
      SUBROUTINE INSERT
C
C------------------------------------------------------------------
C
C      INSERT A LINE INTO THE PROGRAM
C
C      THIS ROUTINE IS PART OF THE PICAX PROGRAM
C      WRITTEN BY ROBERT WALRAVEN, UCD - APPLIED SCIENCE
```

```
C     LAST MODIFICATION ON  4 AUG 80
C
C-----------------------------------------------------------------
C
      COMMON /P C VAR   / NCVAR, CVAR(4)
      COMMON /P INTER   / LINE, ICMND
      COMMON /P MATCH   / MCMND, NMATCH, CHARS, MCHAR2
      COMMON /PROGRM    / NLINES, N LN MAX, PROG(50)
      DIMENSION N(2)
      EQUIVALENCE (PACK,N(1))
      character icmnd*2, chars*2
C
      IF (NLINES .GT. 0) GO TO 10
      NLINES = 1
      INDEX = 1
      GO TO 90
10    DO 20 INDEX = 1,NLINES
      PACK = PROG(INDEX*5-4)
      IF (N(1) .EQ. LINE) GO TO 90
      IF (N(1) .GT. LINE) GO TO 40
20    CONTINUE
      INDEX = NLINES + 1
40    IF (NLINES .LT. N LN MAX) GO TO 60
      CALL ERROR(12)
      RETURN
60    NLINES = NLINES + 1
      IF (INDEX .GE. NLINES) GO TO 90
70    DO 80 J=NLINES,INDEX+1,-1
      M = (J-1)*5
      DO 80 JJ=1,5
      NN = M+JJ
80    PROG (NN) = PROG (NN-5)
90    N(1) = LINE
      N(2) = NMATCH
      PROG(INDEX*5-4) = PACK
      DO 100 I=1,4
100   PROG (INDEX*5+I-4) = CVAR(I)
      RETURN
      END
      SUBROUTINE PLIST
C
C-----------------------------------------------------------------
C
C     LISTS THE CURRENT PROGRAM
C
C     THIS ROUTINE IS PART OF THE PICAX PROGRAM
C     WRITTEN BY ROBERT WALRAVEN, UCD - APPLIED SCIENCE
C     LAST MODIFICATION ON  4 AUG 80
C
C-----------------------------------------------------------------
C
      COMMON /P INTER   / LINE, ICMND
      COMMON /P MATCH   / MCMND, NMATCH, MCHAR1, MCHAR2
      COMMON /PROGRM    / NLINES, N LN MAX, PROG(50)
```

```
      DIMENSION N(2),V(4)
      EQUIVALENCE (PACK,N(1))
      character icmnd*2, mchar1*2
C
      IF (NLINES .EQ. 0) RETURN
      CALL qsmode ( 6 )
      call qcolor (11,0)
      MCMND = 1
      DO 30 I=1,NLINES
      PACK = PROG(I*5-4)
      LINE = N(1)
      nmatch = N(2)
      CALL SEARCH
      ICMND = MCHAR1
      DO 10 J=1,4
10    V(J) = PROG(I*5-4+J)
      WRITE (*,20) LINE,ICMND,V
20    FORMAT(1X,I3,2X,A2,4(1PE15.4))
30    CONTINUE
      MCMND = 0
      LINE = 0
      RETURN
      END
      SUBROUTINE TITLE
C
C-----------------------------------------------------------------------
C
C      GET THE TITLE FROM THE CONSOLE
C
C      THIS ROUTINE IS PART OF THE PICAX PROGRAM
C      WRITTEN BY ROBERT WALRAVEN, UCD - APPLIED SCIENCE
C      LAST MODIFICATION ON  4 AUG 80
C
C-----------------------------------------------------------------------
C
      COMMON /P TITLE  / LTITLE (36)
      character ltitle*2
C
      WRITE (*,10)
10    FORMAT (' TITLE: '$)
      READ (*,20) LTITLE
20    FORMAT(36A2)
      RETURN
      END
C      PICAX3.FOR
C
c
$storage:2
c
c
C-----------------------------------------------------------------------
C
C      PICAX OVERLAY MODULE - REGION 3
C
```

```
C------------------------------------------------------------------
C
C      DESCRIPTION OF SYSTEM COMMANDS
C
C      LAST MODIFICATION ON 17 JUL 81
C
C------------------------------------------------------------------
C
c
c      This program was modifiedfor use with an IBM-PCXT
c      by :
c      Tom Hargreaves
c      JAYCOR
c      205 S. Whiting St.
c      Alexandria, VA 22304
c
c      Last modified on September 26, 1985
c
c------------------------------------------------------------------
c
      SUBROUTINE INFO(N)
      GO TO (100,200,300,400,500,600,700,800,900,1000,1100,1200,1300,
     1   1400,1500,1600,1700,1800,1900,2000,2100,2200,2300,2400,2500,
     2   2600,2700,2800), N
C
100   WRITE (*,110)
110   FORMAT(' EXAMPLE:'/' A 5,1E5'/
     1       ' WILL ADD 1E5 TO VARIABLE 5.')
      RETURN
C
200   CALL QUANAL
      RETURN
C
300   WRITE (*,310)
310   FORMAT(' CLOSE THE DISK OUTPUT FILE.'/'EX: "CL N"'/
     1       ' WILL CLOSE THE FILE ON LOGICAL UNIT N.'/
     2       ' IF N = 0 , THEN THE DEFAULT LUN (2) IS USED.'/
     3       ' IF NO FILE IS OPEN, COMMAND IS IGNORED.')
      RETURN
C
400   WRITE (*,410)
410   FORMAT(' DIRECTORY LISTING:'/' DI'/
     1   ' PRODUCES A DIRECTORY LISTING ON THE CONSOLE.'/
     2   ' DI 6'/' PRODUCES A DIRECTORY LISTING ON THE LINE PRINTER.'/
     3   ' This command is not currently implemented.')
      RETURN
C
500   WRITE (*,510)
510   FORMAT(' DO LOOP:'/' EXAMPLE:'/' DO 30,5'/
     1   ' REPEATS THE SET OF INSTRUCTIONS FROM THE CURRENT LINE TO '/
     2   ' LINE 30 FIVE TIMES.')
      RETURN
C
600   WRITE (*,610)
```

```
610   FORMAT(' ERASES CONSOLE SCREEN.')
      RETURN
C
700   WRITE (*,710)
710   FORMAT(' CLOSES OUTPUT FILE IF OPEN AND EXITS PROGRAM.')
      RETURN
C
800   WRITE (*,810)
810   FORMAT(' FIND AN OLD DISK INPUT FILE.'/' EX: "FI N M"'/
     1       ' WILL CAUSE PROGRAM TO ASK FOR "FILENAME".'/
     2       ' REPLY WITH ANY LEGAL NAME, SUCH AS "datal.dat".'/
     3       ' THE FILE WILL BE OPENED TO LOGICAL UNIT NUMBER N,'/
     4       ' WITH RECORD LENGTH M.  THE DEFAULT VALUES FOR N AND M'/
     5       ' (IF SET TO 0) ARE LUN 3 AND RECORD LENGTH 11700 BYTES.')
      RETURN
C
900   WRITE(*,910)
910   FORMAT(' GO'/' WILL START THE PROGRAM AT THE FIRST LINE.'/
     1       ' GO 25'/' WILL START THE PROGRAM AT LINE 25.')
      RETURN
C
1000  WRITE (*,1010)
1010  FORMAT(' HELP'/' GIVES THE USER A LITTLE USEFUL INFO.')
      RETURN
C
1100  WRITE (*,1110)
1110  FORMAT(' HC'/' ISSUES A COMMAND TO THE HARCOPY UNIT TO'
     1      /' MAKE A COPY OF THE CONSOLE SCREEN.')
      RETURN
C
1200  WRITE (*,1210)
1210  FORMAT(' KILL PROGRAM'/' DELETES THE ENTIRE CURRENT PROGRAM.'/
     1       ' TO DELETE A SINGLE LINE OF THE PROGRAM, TYPE THE LINE'/
     2       ' NUMBER FOLLOWED BY A RETURN.')
      RETURN
C
1300  WRITE (*,1310)
1310  FORMAT(' LIST COMMANDS'/' LIST THE AVAILABLE COMMANDS AND',
     1       ' THEIR BRIEF DESCRIPTION.')
      RETURN
C
1400  WRITE (*,1410)
1410  FORMAT(' LIST PROGRAM'/' LIST THE CURRENT PROGRAM AND'
     1     ' COMMAND VARIABLES.')
      RETURN
C
1500  WRITE (*,1510)
1510  FORMAT(' LIST VARIABLES'/' LIST THE USER VARIABLES,THEIR VALUES,',
     1       ' AND A BRIEF DESCRIPTION.'/
     2       ' EX: "LV N"'/' WILL LIST THE FOLLOWING:'/
     3       ' N       VARIABLES'/
     4       ' 0       EXPERIMENTAL PARAMETERS'/
     5       ' 1       A-D CHANNEL NUMBERS'/
     6       ' 2       TRANSIAC D-A CHANNEL NUMBERS'/
```

```
     7        ' 3        CAMAC SLOT NUMBERS'/
     8        ' 4        TRANSIENT RECORDER ATTENUATOR NUMBERS'/
     9        ' 5        PLOT VARIABLES'/
     1        ' 6        TRANSIENT RECORDER NUMBERS'/
     c        ' 7        DATA TRANSLATION CALIBRATION FACTORS'/
     c        ' 8        TRANSIENT RECORDER CALIBRATION FACTORS'/
     2        ' 10       TRANSIENT RECORDER ATTENUATOR VALUES'/
     3        ' 11       DATA TRANSLATION DATA'/
     4        ' 12       TRANSIENT RECORDER DATA')
      RETURN
C
1600  WRITE (*,1610)
1610  FORMAT(' OPEN A NEW DISK FILE FOR OUTPUT.'/' EX: "O N M"'/
     1        ' WILL CAUSE THE PROGRAM TO ASK FOR A "FILENAME".'/
     2        ' REPLY WITH ANY LEGAL NAME, SUCH AS "data1.dat".'/
     3        ' THE FILE WILL BE OPENED TO LOGICAL UNIT NUMBER N,'/
     4        ' WITH RECORD LENGTH M.  THE DEFAULT VALUES FOR N AND M'/
     5        ' (IF SET TO 0) ARE LUN 2 AND RECORD LENGTH 11700 BYTES.')
      RETURN
C
1700  CALL QUPLOT
      RETURN
C
1800  WRITE (*,1810)
1810  FORMAT(' PAUSE'/' PAUSES UNTIL A RETURN IS TYPED.')
      RETURN
C
1900  CALL QUPROC
      RETURN
C
2000  CALL QUREAD
      RETURN
C
2100  CALL QUSTRT
      RETURN
C
2200  WRITE (*,2210)
2210  FORMAT(' TITLE'/' THE NEXT LINE TYPED WILL BE A TITLE',
     1    ' FOR THE DATA.')
      RETURN
C
2300  WRITE (*,2310)
2310  FORMAT(' EXAMPLE:'/' V3 = 1E5'/' WILL SET USER VARIABLE 3 TO 1E5')
      RETURN
C
2400  CALL QUWRIT
      RETURN
C
2500  WRITE (*,2510)
2510  FORMAT(' EXAMPLE:'/' WAIT 100'/
     1        ' WILL WAIT FOR 100 TIMES APPROXIMATELY 9.2 MILLISECONDS,',
     2        ' OR ABOUT 0.92 SECONDS.')
      RETURN
C
```

```
2600  CALL QUZERO
      RETURN
2700  CALL UINFO (N)
      RETURN
C
C
2800  RETURN
      END
```

```
*********************************************************************
c
c      userr.qo
c
c      This module was written to interface PICAX with
c      the Quasi-optical gyrotron experiment at the
c      Naval Research Lab, Washington D.C.
c
c      Written by:
c            Tom Hargreaves
c            JAYCOR
c            205 S. Whiting St.
c            Alexandria, VA 22304
c
c      Last modification:     September 27, 1985
c
c*********************************************************************
c
c
$storage:2
c
c--------------------------------------------------------------------
      SUBROUTINE UPLOT
c--------------------------------------------------------------------
c
c       This routine plots the data.
c
c       cvar (1) = 0    Cathode voltage vs. time
c                  1    Intermediate voltage vs. time
c                  2    Cathode current vs. time
c                  3    Collector current vs. time
c                  4    Microwave diode vs. time
c                  5    Interferometer diode vs. time
c                  6    Interferometer diode vs. mirror spacing
c                  7    Microwave diode vs. cathode voltage
c                  8    Microwave diode vs. cathode current
c                  9    Microwave diode vs. collector current
c                 10    Microwave diode vs. magnetic field
c
c--------------------------------------------------------------------
c
      common / p c var / n c var, c var (4)
      COMMON /U VAR   / N U VAR, U VAR (150)
      common / data / atten (20), datran (16,2), trrec (1000,10),
     c         freq (1000,2), data (100,10)
      common / stp num / n stp b, n stp tp, n stp v, n stp cr,
     c         n stp al, n dat, n freq, n dat ar
      common / datime / idate (4), itime (4)
      dimension time (1000)
      dimension y1(1000), y2(1000), y3(1000), y4(1000), y5(1000)
      dimension err1(1000), err2(1000), err3(1000), err4(1000),
     c       err5(1000)
      equivalence (y1(1),trrec(1,1)),(err1(1),trrec(1,2)),
     c               (y2(1),trrec(1,3)),(err2(1),trrec(1,4)),
```

```
     c              (y3(1),trrec(1,5)),(err3(1),trrec(1,6)),
     c              (y4(1),trrec(1,7)),(err4(1),trrec(1,8)),
     c              (y5(1),trrec(1,9)),(err5(1),trrec(1,10))
      dimension freql (1000), pos (1000)
      equivalence (freql(1),freq(1,2)),(pos(1),freq(1,1))
      dimension y6(100), y7(100), y8(100), y9(100), y10(100)
      dimension err6(100), err7(100), err8(100), err9(100), err10(100)
      equivalence (y6(1),data(1,1)),(err6(1),data(1,2)),
     c            (y7(1),data(1,3)),(err7(1),data(1,4)),
     c            (y8(1),data(1,5)),(err8(1),data(1,6)),
     c            (y9(1),data(1,7)),(err9(1),data(1,8)),
     c            (y10(1),data(1,9)),(err10(1),data(1,10))
      character xlabel*25, ylabel*25
c
c       For the x-axis label :  jcolx = 340 - 4 * ncharx
c                               jrowx = 10
c       For the y-axis label :  jcoly = 8
c                               jrowy = 100 + 4 * nchary
c       For the title :         jcolt = 340 - 4 * nchart
c                               jrowt = 189
c
        if(cvar(1) .ge. 0.0 .and. cvar(1) .le. 5.0) then
c
c       Calculate the time array (in microseconds).
c
          deltat = 1.0 / 30.0
          do 10 i = 1,1000
10        time (i) = float (i-1) * deltat
          nplot = 101 + ifix (cvar(1))
          nplot = ifix (uvar(nplot))
          npt = ifix (uvar(17 + nplot))
           if (nplot .eq. 0) then
            write (*,'('' No data for plot.'')')
            return
           elseif (nplot .eq. 1) then
            call uplotl (time,y1,npt)
           elseif (nplot .eq. 2) then
            call uplotl (time,y2,npt)
           elseif (nplot .eq. 3) then
            call uplotl (time,y3,npt)
           elseif (nplot .eq. 4) then
            call uplotl (time,y4,npt)
           elseif (nplot .eq. 5) then
            call uplotl (time,y5,npt)
           else
            write (*,'('' Illegal transient recorder number.'')')
            return
           endif
         elseif (cvar(1) .eq. 6.0) then
          npt = int (uvar(16))
          call uplotl (pos,freql,npt)
         elseif (cvar(1) .ge. 7.0 .and. cvar(1) .le. 10.0) then
          npt = n dat ar
           if (cvar(1) .eq. 7.0) then
```

```
          call uplotl (y7,y6,npt)
         elseif (cvar(1) .eq. 8.0) then
          call uplotl (y8,y6,npt)
         elseif (cvar(1) .eq. 9.0) then
          call uplotl (y9,y6,npt)
         elseif (cvar(1) .eq. 10.0) then
          call uplotl (y10,y6,npt)
         endif
       else
        write (*,'('' Plot number out of range.'')')
        return
       endif
c
c      Do axis labeling here.
c
      if (cvar(1) .ge. 0.0 .and. cvar(1) .le. 5.0) then
       xlabel = 'Time  (microsec)'
       ncharx = 16
       jcolx = 276
      elseif (cvar(1) .eq. 6.0) then
       xlabel = 'Mirror Spacing  (mm)'
       ncharx = 20
       jcolx = 260
      elseif (cvar(1) .ge. 7.0 .and. cvar(1) .le. 10.0)then
       ylabel = 'Microwave diode'
       nchary = 15
       jrowy = 160
      endif
      if (cvar(1) .eq. 0.0) then
       ylabel = 'Cathode V  (kV)'
       nchary = 15
       jrowy = 160
      elseif (cvar(1) .eq. 1.0) then
       ylabel = 'Intermediate V  (kV)'
       nchary = 20
       jrowy = 180
      elseif (cvar(1) .eq. 2.0) then
       ylabel = 'Cathode I  (A)'
       nchary = 14
       jrowy = 156
      elseif (cvar(1) .eq. 3.0) then
       ylabel = 'Collector I  (A)'
       nchary = 16
       jrowy = 164
      elseif (cvar(1) .eq. 4.0) then
       ylabel = 'Microwave diode'
       nchary = 15
       jrowy = 160
      elseif (cvar(1) .eq. 5.0) then
       ylabel = 'Interferometer diode'
       nchary = 20
       jrowy = 180
      elseif (cvar(1) .eq. 6.0) then
       ylabel = 'Interferometer Diode'
```

```
           nchary = 20
           jrowy = 180
          elseif (cvar(1) .eq. 7.0) then
           xlabel = 'Cathode V  (kV)'
           ncharx = 15
           jcolx = 280
          elseif (cvar(1) .eq. 8.0) then
           xlabel = 'Cathode I  (A)'
           ncharx = 14
           jcolx = 284
          elseif (cvar(1) .eq. 9.0) then
           xlabel = 'Collector I  (A)'
           ncharx = 16
           jcolx = 276
          elseif (cvar(1) .eq. 10.0) then
           xlabel = 'Magnetic Field  (kG)'
           ncharx = 20
           jcolx = 260
          endif
         call qgtxt (ncharx,xlabel,3,jcolx,10,0)
         call qgtxt (nchary,ylabel,3,8,jrowy,-1)
c
c        Print the time and date that the data was taken.
c
         write (*,'(57xi2''/''i2''/''i4'' , ''i2'':''i2'':''i2)')
        c   idate(2), idate(3), idate(1), (itime(i), i = 1,3)
c
         return
         end
c
         subroutine uplotl ( x, y, npt )
c
c---------------------------------------------------------------------------
c
c          This routine does the actual plotting as well as
c          drawing both sets of axis.
c
c---------------------------------------------------------------------------
c
           dimension x(1000), y(1000)
           common / p c var / n c var, c var (4)
           COMMON /U VAR   / N U VAR, U VAR (150)
c
c          Check for xst, xfin, yst, and yfin (the plot boundaries).
c
           xst = uvar (97)
           xfin = uvar (98)
           call minmax (xst,xfin,x,npt,xmajor,nx)
           yst = uvar (99)
           yfin = uvar (100)
           call minmax (yst,yfin,y,npt,ymajor,ny)
c
c          Calculate xmin, xmax, ymin, ymax.
c
```

```
      xmin = xst - 0.17 * (xfin - xst)
      xmax = xfin + 0.1 * (xfin - xst)
      ymin = yst - 0.23 * (yfin - yst)
      ymax = yfin + 0.11 * (yfin - yst)
c
c     Set up for the plot.
c
      iopt = 0
      call qplot (0,639,0,199,xmin,xmax,ymin,ymax,xst,yst,
     c   iopt,1.0,1.0)
      if (iopt .eq. -2) then
      write (*,'('' Input error'')')
      return
      endif
      call qsetup (0,3,-1,3)
c
c     Plot the points here.
c
      call qtabl (0,npt,x,y)
c
c     Plot the first set of axis here.
c
      call qxaxis (xst,xfin,xmajor,1,1,nx)
      call qyaxis (yst,yfin,ymajor,1,1,ny)
c
c     Plot the second set of axis here.
c
      iopt = 0
      call qplot (0,639,0,199,xmin,xmax,ymin,ymax,xfin,yfin,
     c   iopt,1.0,1.0)
      if (iopt .eq. -2) then
      write (*,'('' Input error for the second set of axis.'')')
      return
      endif
      xmajor = - xmajor
      ymajor = - ymajor
      call qxaxis (xst,xfin,xmajor,1,0,0)
      call qyaxis (yst,yfin,ymajor,1,0,0)
c
      return
      end
c
      subroutine minmax (rst,rfin,r,npt,rmajor,n)
c
c-----------------------------------------------------------------
c
c     This routine finds the minimum and maximum of
c     the array r. It also calculates some of the
c     parameters used in the plotting routines.
c
c-----------------------------------------------------------------
c
      dimension r (1000)
c
```

```
       if (rst .eq. rfin) then
        rst = amin1 (r(1),r(2))
        rfin = amax1 (r(1),r(2))
        do 10 i = 3,npt
        rst = amin1 (rst,r(i))
10      rfin = amax1 (rfin,r(i))
        dif = rfin - rst
        rst = rst - 0.1 * dif
        rfin = rfin + 0.1 * dif
       endif
       if (rst .eq. rfin) then
        if (rst .ne. 0.0) then
         rst = 0.95 * rst
         rfin = 1.05 * rfin
        else
         rst = -1.0
         rfin = 1.0
        endif
       endif
c
c     We need 1.0e6 > rst,rfin > -1.0e6
c
       if (rst .gt. 1.0e6) rst = 1.0e6 - 1.0
       if (rst .lt. -1.0e6) rst = -1.0e6
       if (rfin .gt. 1.0e6) rfin = 1.0e6
       if (rfin .lt. -1.0e6) rfin = -1.0e6 + 1.0
c
      dif = abs (rfin - rst)
c
c       Calculate the distance between the major tic marks
c       and pick n for the label format f10.n.
c
      tl = 10.0
      n = 0
       if (dif .gt. 5.0) then
        do 20 i = 1,5
         if (dif .le. tl) then
          rmajor = 0.1 * tl
          go to 40
         elseif (dif .le. (2.0 * tl)) then
          rmajor = 0.2 * tl
          go to 40
         elseif (dif .le. (5.0 * tl)) then
          rmajor = 0.5 * tl
          go to 40
         endif
20      tl = tl * 10.0
        write (*,'('' Cannot compute the tic mark spacing.'')')
        rmajor = tl
        go to 40
       endif
      n = 1
      tl = 1.0
      do 30 i = 1,8
```

```
        if (dif .gt. (5.0 * t1)) then
         rmajor = t1
         go to 40
        elseif (dif .gt. (2.0 * t1)) then
         rmajor = 0.5 * t1
         go to 40
        elseif (dif .gt. t1) then
         rmajor = 0.2 * t1
         go to 40
        endif
       n = n + 1
30     t1 = t1 * 0.1
       write (*,'('' Cannot compute the tic mark spacing.'')')
       rmajor = t1
40     continue
c
c     We need 0 <= n <= 3
c
        if (n .gt. 3) n = 3
c
       rst = anint ((rst / rmajor)) * rmajor
       rfin = anint ((rfin / rmajor)) * rmajor
c
       return
       end
```

```
C***********************************************************************
C
c       user1.qo
c       This module was written to interface PICAX with
c       the Quasi-optical gyrotron experiment at the
c       Naval Research Lab, Washington D.C.
c
c       Written by:
c             Tom Hargreaves
c             JAYCOR
c             205 S. Whiting St.
c             Alexandria, VA 22304
c
c       Last modification:  October 3, 1985
C
C***********************************************************************
c
c
$storage:2
c
C
      SUBROUTINE UANLYZ
c
c-----------------------------------------------------------------------
c
c     This routine will calculate the peak power as well as the
c     efficiency using the calorimeter data as well as the pulse
c     shape, voltage, and current data from the transient recorders.
c     If the transient recorder data is unavailable, the routine will
c     prompt for the pulse width, and will use the voltage and current
c     data from the Data Translation card.
c
c     cvar (1) = 0     Prompt user for rep rate.
c              > 0     Rep rate.
c
c-----------------------------------------------------------------------
c
      COMMON /U VAR   / N U VAR, U VAR (150)
      common / p c var  / n c var, cvar (4)
      common / data / atten (20), datran (16,2), trrec (1000,10),
     c          freq (1000,2), data (100,10)
c
c     Calculate the area under the microwave diode curve and the
c     corresponding pulse width for the peak power.  If the data
c     is not available from the transient recorder, then ask the
c     user for the pulse width.
c
c     Note: It is assumed that the diode is working in the linear
c           regime.
c
      p width = 0.0
      p max = 0.0
      total = 0.0
      n max = 0
```

```
      if (uvar (105) .eq. 0.0) then
       write (*,'('' The microwave diode is unavailable from the ''
     c   ''transient recorder.''/'' Enter the pulse width (sec) : ''$)')
       read (*,*) p width
      else
       n = int (uvar (105) * 2.0 - 1.0)
       do 10 i = 1,int (uvar (17+int (uvar (105))))
       p max = amax1 (p max,tr rec(i,n))
       if (p max .eq. tr rec (i,n)) n max = i
10     total = total + tr rec (i,n)
c
c    Assume that the transient recorders are running at 30 MHz.
c    Therefor the points are 33.3 nsec apart.
c
       if (p max .ne. 0.0) p width = 33.3e-9 * total / p max
      endif
c
c    Now get the rep rate.  If cvar (1) = 0, prompt the user.
c
      rep rat = cvar (1)
      if (rep rat .eq. 0.0) then
       write (*,'('' Enter rep rate (Hz) : ''$)')
       read (*,*) rep rat
      endif
c
c    Calculate the peak power (in kW) from the average power (in watts)
c    of the calorimeter.
c
      p peak = 0.0
      if(rep rat .ne. 0.0 .and. p width .ne. 0.0) then
       p peak = datran(int (uvar (40)),1)/(rep rat * p width * 1000.0)
      endif
c
c    Get the cathode voltage from the transient recorder if it is
c    available, otherwise use the Data Translation data.
c
      volt = 0.0
      n volt = 0
      if (uvar (101) .ne. 0.0 .and. n max .ne. 0 .and.
     c   int(uvar(17+int(uvar(101)))) .ge. n max) then
       n volt = int (2.0 * uvar (101) - 1.0)
       volt = tr rec (n max,n volt)
      else
       volt = datran (int (uvar (35)),1)
      endif
c
c    Get the collector current from the transient recorder if it is
c    available, otherwise use the Data Translation data.
c
      cur = 0.0
      n cur = 0
      if (uvar (103) .ne. 0.0 .and. n max .ne. 0 .and.
     c   int(uvar(17+int(uvar(103)))) .ge. n max) then
       n cur = int (2.0 * uvar (103) - 1.0)
```

```
       cur = tr rec (n max,n cur)
      else
       cur = datran (int (uvar (38)),1)
      endif
c
c    Calculate the efficiency here.
c
      eta = 0.0
      if (cur .ne. 0.0 .and. volt .ne. 0.0) then
       eta = p peak / (volt * cur) * 100.0
      endif
c
c    Now print out the results.
c
      write (*,'(/
     c    '' Diode pulse height = ''1pe10.3''  volts''/
     c    '' Pulse width = ''1pe10.3''  seconds''/
     c    '' Rep rate = ''1pe10.3''  Hz''/
     c    '' Cathode voltage = ''1pe10.3''  kV''/
     c    '' Collector current = ''1pe10.3''  Amps'')')
     c   p max, p width, rep rat, volt, cur
      write (*,'(/'' Peak power = ''1pe10.3''  kW''/
     c    '' Efficiency = ''1pe10.3''  %'')') p peak, eta
      RETURN
      END
c
C-----------------------------------------------------------------
      SUBROUTINE UINIT
c
c-----------------------------------------------------------------
c
c      This routine initializes the variables from the
c      user written subroutines.
c
c-----------------------------------------------------------------
c
      COMMON /U VAR   / N U VAR, U VAR (150)
      common/p files/nopen,lunopn(6),lunfnd(6),nfound,
     c                 irecl(6),irnr(6),irnw(6),irnwmx(6)
      common / data / atten (20), datran (16,2), trrec (1000,10),
     c         freq (1000,2), data (100,10)
      common / stp num / n stp b, n stp tp, n stp v, n stp cr,
     c         n stp al, n dat, n freq, n dat ar
      common / increm / del b, del tp, del v, del cr, del al
      common / u flags / strt fl, freq fl, base fl, n dt err, dt err
      logical strt fl, freq fl, base fl, dt err
      common / DT 2801 / base ad, com reg, stat rg, dat reg
      integer base ad, com reg, stat rg, dat reg
      common / datime / idate (4), itime (4)
c
c     The date and time arrays.
c
      do 5 i = 1,4
      idate (i) = 0
```

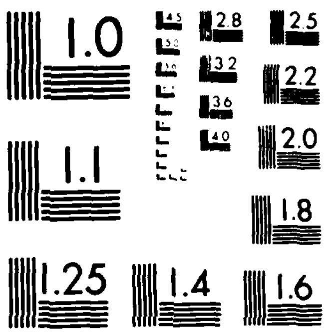
```
5     itime (i) = 0
c
c      The user variables.
c
      N U VAR = 150
      do 10 i=1,nuvar
10    U VAR (i) = 0.0
c
c      The read and write variables.
c
      do 20 i=1,6
      irecl (i) = 11700
      irnr (i) = 1
      irnwmx (i) = 0
20    irnw (i) = 1
c
c      The data arrays.
c
      do 30 i=1,20
30    atten (i) = 0.0
      do 40 i=1,16
      do 40 j=1,2
40    datran (i,j) = 0.0
      do 50 i=1,1000
      do 50 j=1,10
50    trrec (i,j) = 0.0
      do 60 i = 1,1000
      do 60 j = 1,2
60    freq (i,j) = 0.0
      do 70 i = 1,100
      do 70 j = 1,10
70    data (i,j) = 0.0
c
c      The data step numbers.
c
      n stp b = 1
      n stp tp = 1
      n stp v = 1
      n stp cr = 1
      n stp al = 1
      n dat = 0
      n freq = 1
      n dat ar = 0
c
c      The data increments.
c
      del b = 0.0
      del tp = 0.0
      del v = 0.0
      del cr = 0.0
      del al = 0.0
c
c      The user flags.
c
```

```
      strt fl = .false.
      freq fl = .false.
      base fl = .true.
c
c      Set the crate number to 1, and initialize the crate.
c
      call crate (1)
      call camcl (1)
c
c     Set up the crate for DMA input and output.
c
      call dmaset (1,2,1,1000)
c
c     Start the Data Translation board sampling.
c
      base ad = #2ec
      com reg = base ad + 1
      stat rg = base ad + 1
      dat reg = base ad
c
c     Stop the DT2801 board read any junk data and then clear any errors.
c
      call out (com reg, #f)
c
      junk = inp (dat reg)
      call waitl (stat rg, #4, 0)
      call out (com reg, #1)
c
c     Set the clock rate. Send the command byte first.
c
      call waitl (stat rg, 4, 0)
      call out (com reg, 3)
c
c     Send the clock period = 250 microseconds = 2.5 microsec * 100
c     Note that the clock period is sent as two bytes.
c     Note that 99 * 2.5 microsec is the minimum period that allows
c     the subroutine dtinp.asm to collect the data.
c
      call waitl (stat rg, 2, 2)
      call out (dat reg, 100)
      call waitl (stat rg, 2, 2)
      call out (dat reg, 0)
c
c     Set the A-D parameters. Send the command byte first.
c
      call waitl (stat rg, 4, 0)
      call out (com reg, #d)
c
c     Send the gain code = 0
c
      call waitl (stat rg, 2, 2)
      call out (dat reg, 0)
c
c     Send the A-D start channel = 0.
```

```
c
      call waitl (stat rg, 2, 2)
      call out (dat reg, 0)
c
c     Send the A-D end channel = 15
c
      call waitl (stat rg, 2, 2)
      call out (dat reg, 15)
c
c     Send the number of A-D conversions = 16.
c     Note that two bytes are sent.
c
      call waitl (stat rg, 2, 2)
      call out (dat reg, 16)
      call waitl (stat rg, 2, 2)
      call out (dat reg, 0)
c
c     Set digital port 0 for output. Send the command byte first.
c
      call waitl (stat rg, 4, 0)
      call out (com reg, 5)
      call waitl (stat rg, 2, 2)
      call out (dat reg, 0)
c
c     Set all bits to 0 for port 0. Send the command byte first.
c
      call waitl (stat rg, 4, 0)
      call out (com reg, 7)
      call waitl (stat rg, 2, 2)
      call out (dat reg, 0)
      call waitl (stat rg, 2, 2)
      call out (dat reg, 0)
c
c     Check for any errors.
c
      call waitl (stat rg, 4, 0)
      istat = inp (stat rg)
      if (istat .ge. #80) then
c
c     Stop the DT2801 and read any junk data.
c
        call out (com reg, #f)
        junk = inp (dat reg)
c
c     Send the read error register command.
c
        call waitl (stat rg, 4, 0)
        call out (com reg, 2)
c
c     Read the error register. Note that there are two bytes.
c
        call waitl (stat rg, 5, 0)
        ierrl = inp (dat reg)
        call waitl (stat rg, 5, 0)
```

AD-A171 873 A SYSTEM FOR INTERACTIVE COMPUTER CONTROL OF EXPERIMENTS(U) NAVAL RESEARCH LAB WASHINGTON DC T A HARGREAVES ET AL. 25 AUG 86 NRL-MR-5743 2/2

UNCLASSIFIED F/G 9/2 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

```
      ierrh = inp (dat reg)
c
      write(*,'('' Error initializing the Data Translation board.'')')
      write (*,'('' DT2801 error register low byte = ''i3)') ierrl
      write (*,'('' DT2801 error register high byte = ''i3)') ierrh
      endif
c
      RETURN
      END
c
C-----------------------------------------------------------------------
c
      SUBROUTINE UREAD (LUN,lnum)
c
c-----------------------------------------------------------------------
c
c      This routine reads record number irnr (lnum) from
c      logical unit number lun.
c
c      cvar(1) = 0      lun = 3
c              = lun
c
c      cvar(2) = 0      irnr(lnum) = next record
c              = irnr (lnum)
c
c-----------------------------------------------------------------------
c
      common/p files/nopen,lunopn(6),lunfnd(6),nfound,
     c                irecl(6),irnr(6),irnw(6),irnwmx(6)
      common / p c var  / n c var, cvar (4)
      COMMON /U VAR   / N U VAR, U VAR (150)
      common / data / atten (20), datran (16,2), trrec (1000,10),
     c         freq (1000,2), data (100,10)
      common / stp num / n stp b, n stp tp, n stp v, n stp cr,
     c         n stp al, n dat, n freq, n dat ar
      common / p title / l title (36)
      character l title*2
      common / datime / idate (4), itime (4)
c
      if (cvar(2) .ne. 0.0) irnr(lnum) = cvar(2)
      read(lun,rec=irnr(lnum),end=100,err=110)irecln,
     c    nuvar, (ltitle(i),i=1,36), (idate(i),i=1,4), (itime(i),i=1,4),
     c     (uvar(i),i=1,nuvar),
     c     (atten(i),i=1,20),
     c     ((datran(i,j),i=1,16),j=1,2),
     c     ((trrec(i,j),i=1,int(uvar(17+((1+j)/2)))),j=1,10),
     c     ((freq(i,j),j=1,2),i=1,int(uvar(16)))
      write(*,'('' Record number ''i3'' is ''i6'' bytes long.'')')
     c       irnr (lnum), irecln
      irnr(lnum) = irnr(lnum) + 1
c
c    Put data into the accumulative data arrays.
c
      n dat ar = n dat ar + 1
```

```
      data (n dat ar,1) = datran (int(uvar(39)),1)
      data (n dat ar,2) = datran (int(uvar(39)),2)
      data (n dat ar,3) = datran (int(uvar(35)),1)
      data (n dat ar,4) = datran (int(uvar(35)),1)
      data (n dat ar,5) = datran (int(uvar(37)),1)
      data (n dat ar,6) = datran (int(uvar(37)),1)
      data (n dat ar,7) = datran (int(uvar(38)),1)
      data (n dat ar,8) = datran (int(uvar(38)),1)
      data (n dat ar,9) = 25.0 * (datran (int(uvar(31)),1) / 118.8
     c    + datran (int(uvar(32)),1) / 121.4)
      data (n dat ar,10) = 25.0 * (datran (int(uvar(31)),2) / 118.8
     c    + datran (int(uvar(32)),2) / 121.4)
      RETURN
100   write (*,'('' Attempt to read past end of file'')')
      return
110   write(*,'('' Error on read'')')
c
      RETURN
      END
c
C------------------------------------------------------------------
c
      SUBROUTINE UWRITE (LUN,lnum)
c
c------------------------------------------------------------------
c
c      This routine writes record number irnw(lnum)
c      of length irecl to logical unit number lun.
c
c      cvar(1) = 0      lun = 2
c              = lun
c
c      cvar(2) = 0      irnw = next record
c              = irnw
c
c------------------------------------------------------------------
c
      common/p files/nopen,lunopn(6),lunfnd(6),nfound,
     c                irecl(6),irnr(6),irnw(6),irnwmx(6)
      common / p c var  / n c var, cvar (4)
      COMMON /U VAR    / N U VAR, U VAR (150)
      common / data / atten (20), datran (16,2), trrec (1000,10),
     c         freq (1000,2), data (100,10)
      common / stp num / n stp b, n stp tp, n stp v, n stp cr,
     c         n stp al, n dat, n freq, n dat ar
      common / p title / l title (36)
      character l title*2
      common / datime / idate (4), itime (4)
c
      if (int(cvar(2)) .gt. (irnwmx(lnum)+1)) then
      write(*,'('' Out of sequence write is not allowed here.''/
     c     '' No record was written.'')')
      return
      elseif(cvar(2).ne.0.0.and.int(cvar(2)).ne.(irnwmx(lnum)+1))then
```

```
20    write(*,'('' This will overwrite record number ''i3'' .''
     c      /'' Enter 1 to overwrite, 0 to abort write.'')')int(cvar(2))
      read (*,*) test
      if (test .eq. 0.0) return
      if (test .ne. 1.0) go to 20
      irnw(lnum) = cvar(2)
      else
      irnwmx (lnum) = irnwmx (lnum) + 1
      endif
c
c      Check to see if the data will fit in one record length.
c
      nbytes = 734
      do 100 i=1,5
100   nbytes = nbytes + 8*int(uvar(17+i))
      nbneed = nbytes - irecl (lnum)
      if (nbneed .gt. 0) then
      write(*,'('' Attempt to write past end of record.''/
     c        '' No data was written.''/
     c        1x,i6'' total bytes are needed.'')') nbytes
      return
      endif
c
c      Write data here.
c
      write(lun,err=1000,rec=irnw(lnum))irecl(lnum),
     c    nuvar, (ltitle(i),i=1,36), (idate(i),i=1,4), (itime(i),i=1,4),
     c     (uvar(i),i=1,nuvar),
     c     (atten(i),i=1,20),
     c     ((datran(i,j),i=1,16),j=1,2),
     c     ((trrec(i,j),i=1,int(uvar(17+((1+j)/2)))),j=1,10),
     c     ((freq(i,j),j=1,2),i=1,int(uvar(16)))
      write(*,'('' Record number ''i3'' is ''i6'' bytes long'')')
     c       irnw(lnum), nbytes
      irnw (lnum) = irnwmx (lnum) + 1
      RETURN
1000  WRITE (*,*) 'Error on write'
c
      RETURN
      END
c
C-----------------------------------------------------------------------
      SUBROUTINE UZERO
      common / data / atten (20), datran (16,2), trrec (1000,10),
     c          freq (1000,2), data (100,10)
      common / stp num / n stp b, n stp tp, n stp v, n stp cr,
     c          n stp al, n dat, n freq, n dat ar
c
c     Zero the accumulative data array here.
c
      do 10 i = 1,100
      do 10 j = 1,10
10    data (i,j) = 0.0
c
      n dat ar = 0
c
      RETURN
      END
```

```
C*************************************************************************
C
c       user2.qo
c
c       This module was written to interface PICAX with
c       the Quasi-optical gyrotron experiment at the
c       Naval Research Lab, Washington D.C.
c
c       Written by:
c             Tom Hargreaves
c             JAYCOR
c             205 S. Whiting St.
c             Alexandria, VA 22304
c
c       Last modification:     October 3, 1985
C
C*************************************************************************
c
c
$storage:2
c
C
      SUBROUTINE U MATCH
c
c-------------------------------------------------------------------------
c
c    This routine defines additional (user written) commands.
c
c-------------------------------------------------------------------------
c
      logical match
c
      if(match(27,'tr  ','Trigger on/off                    '))return
c
      RETURN
      END
c
      SUBROUTINE UCMNDS (N)
c
C-------------------------------------------------------------------------
c
c    The user written commands are executed here.
c
c-------------------------------------------------------------------------
c
      common /p c var/ n c var, c var (4)
      common / DT 2801 / base ad, com reg, stat rg, dat reg
      integer base ad, com reg, stat rg, dat reg
c
c-----------------------------------------Trigger on/off
c
      if (n .eq. 27) then
       i data = int (cvar(1))
c
```

```
c    Send the data to port 0. Send the command byte first.                44
c
        call waitl (stat rg, 4, 0)
        call out (com reg, 7)
        call waitl (stat rg, 2, 2)
        call out (dat reg, 0)
        call waitl (stat rg, 2, 2)
        call out (dat reg, i data)
        return
        endif
c
      RETURN
      END
c
      SUBROUTINE UINFO (N)
c
C------------------------------------------------------------------
c
c    This routine gives information about the user written commands.
c
c------------------------------------------------------------------
c
      if (n .eq. 27) then
        write (*,270)
270     format (' Trigger on/off'//' ex:  "tr n"  will:'//
     1   ' n = 0      Turn trigger off.'/
     2   ' n = 1      Turn trigger on.')
        return
        endif
c
      RETURN
      END

c------------------------------------------------------------------
      SUBROUTINE UVLIST
c
c------------------------------------------------------------------
c
c       This subroutine lists the user variables.
c       Different variables are listed depending on the value
c       of variable cvar(1).
c
c       cvar(1) = 0      List variables numbered 1-30
c               = 1      List variables numbered 31-46
c               = 2      List variables numbered 47-62
c               = 3      List variables numbered 63-75
c               = 4      List variables numbered 76-96
c               = 5      List variables numbered 97-100
c               = 6      List variables numbered 101-110
c               = 7      List variables numbered 111-130
c               = 8      List variables numbered 131-140
c               = 10     List attenuator values
c               = 11     List datran (i,j)
c               = 12     List trrec (i,j) for channel cvar(2)
```

```
c
c---------------------------------------------------------------------
c
      common / p c var / n c var, c var (4)
      COMMON /U VAR   / N U VAR, U VAR (150)
      common / data / atten (20), datran (16,2), trrec (1000,10),
     c         freq (1000,2), data (100,10)
      common / p title / l title (36)
      character type*1, string*35, l title*2
      common / datime / idate (4), itime (4)
c
      write(*,'('' Title: '',36a2)') l title
      write (*,'('' Experiment date and time :  ''i2''/''i2''/''i4
     c  '' , ''i2'':''i2'':''i2)') idate (2), idate (3), idate (1),
     c  (itime(i), i = 1,3)
      n = 1 + cvar (1)
      go to (10,20,30,40,50,60,70,80,90,1,100,110,120) n
1     write(*,'('' Parameter out of range.'')')
      call info (15)
      return
c
c      List experimental parameters here.
c
10    write(*,'('' List of experimental parameters.'')')
      type = 'r'
      string = 'Initial cavity magnetic field (kG).'
      call define (1,type,string)
      string = 'Final cavity magnetic field (kG).'
      call define (2,type,string)
      type = 'i'
      string = 'Number of steps.'
      call define (3,type,string)
      type = 'r'
      string = 'Initial cavity field taper (%).'
      call define (4,type,string)
      string = 'Final cavity field taper (%).'
      call define (5,type,string)
      type = 'i'
      string = 'Number of steps.'
      call define (6,type,string)
      type = 'r'
      string = 'Initial e-beam voltage (kV).'
      call define (7,type,string)
      string = 'Final e-beam voltage (kV).'
      call define (8,type,string)
      type = 'i'
      string = 'Number of steps.'
      call define (9,type,string)
      type = 'r'
      string = 'Initial e-beam current (A).'
      call define (10,type,string)
      string = 'Final e-beam current (A).'
      call define (11,type,string)
      type = 'i'
```

```
      string = 'Number of steps.'
      call define (12,type,string)
      type = 'r'
      string = 'Initial e-beam alpha.'
      call define (13,type,string)
      string = 'Final e-beam alpha.'
      call define (14,type,string)
      type = 'i'
      string = 'Number of steps.'
      call define (15,type,string)
      string = 'Number of interferometer points.'
      call define (16,type,string)
      string = 'Number of shots/pt. to avg. over.'
      call define (17,type,string)
      string = 'Number of pts. for tran. rec. 1.'
      call define (18,type,string)
      string = 'Number of pts. for tran. rec. 2.'
      call define (19,type,string)
      string = 'Number of pts. for tran. rec. 3.'
      call define (20,type,string)
      string = 'Number of pts. for tran. rec. 4.'
      call define (21,type,string)
      string = 'Number of pts. for tran. rec. 5.'
      call define (22,type,string)
      return
c
c      List analog-digital channel numbers here.
c
20    write(*,'('' List of Data Translation channel numbers.'')')
      type = 'i'
      string = 'Supercon top coil current.'
      call define (31,type,string)
      string = 'Supercon bottom coil current.'
      call define (32,type,string)
      string = 'Trim top coil current.'
      call define (33,type,string)
      string = 'Trim bottom coil current.'
      call define (34,type,string)
      string = 'Cathode voltage.'
      call define (35,type,string)
      string = 'Intermediate anode voltage.'
      call define (36,type,string)
      string = 'Cathode current.'
      call define (37,type,string)
      string = 'Collector current.'
      call define (38,type,string)
      string = 'Microwave diode.'
      call define (39,type,string)
      string = 'Calorimeter.'
      call define (40,type,string)
      string = 'Interferometer position.'
      call define (41,type,string)
      string = 'Interferometer diode.'
      call define (42,type,string)
```

```
      return
c
c      List Transiac digital-analog channel numbers here.
c
30    write(*,'('' Transiac digital-analog channel numbers.'')')
      type = 'i'
      string = 'Supercon top coil current.'
      call define (47,type,string)
      string = 'Supercon bottom coil current.'
      call define (48,type,string)
      string = 'Trim top coil current.'
      call define (49,type,string)
      string = 'Trim bottom coil current.'
      call define (50,type,string)
      string = 'Cathode current.'
      call define (51,type,string)
      string = 'Collector current.'
      call define (52,type,string)
      return
c
c      List Camac slot numbers here.
c
40    write(*,'('' Camac slot numbers.'')')
      type = 'i'
      string = 'Controller.'
      call define (63,type,string)
      string = 'Digital-analog converter.'
      call define (64,type,string)
      string = 'Hex attenuator.'
      call define (65,type,string)
      string = 'Differential amplifier.'
      call define (66,type,string)
      string = 'Transient recorder number 1.'
      call define (67,type,string)
      string = 'Transient recorder number 2.'
      call define (68,type,string)
      string = 'Transient recorder number 3.'
      call define (69,type,string)
      string = 'Transient recorder number 4.'
      call define (70,type,string)
      string = 'Transient recorder number 5.'
      call define (71,type,string)
      return
c
c      List the attenuator numbers for the transient recorders here.
c      Hex attenuators are numbers 1-6.
c      Differential amplifier is number 7.
c
50    write(*,'('' Attenuator numbers (Differential amp is 7)'')')
      write(*,'(/'' Note: These are applied only to the transient''
     c    '' recorder data.'')')
      type = 'i'
      string = 'Number of attenuators (20 maximum).'
      call define (76,type,string)
```

```
      string = 'Superconductor top coil current.'
      call define (77,type,string)
      string = 'Superconductor bottom coil current.'
      call define (78,type,string)
      string = 'Top trim coil current.'
      call define (79,type,string)
      string = 'Bottom trim coil current.'
      call define (80,type,string)
      string = 'Cathode voltage.'
      call define (81,type,string)
      string = 'Intermediate anode voltage.'
      call define (82,type,string)
      string = 'Cathode current.'
      call define (83,type,string)
      string = 'Collector current.'
      call define (84,type,string)
      string = 'Microwave diode.'
      call define (85,type,string)
      string = 'Calorimeter.'
      call define (86,type,string)
      string = 'Interferometer position.'
      call define (87,type,string)
      string = 'Interferometer diode.'
      call define (88,type,string)
      return
c
c      List plot variables here.
c
60    Write(*,'('' Plot variables.'')')
      type = 'r'
      string = 'X-axis minimum.'
      call define (97,type,string)
      string = 'X-axis maximum.'
      call define (98,type,string)
      string = 'Y-axis minimum.'
      call define (99,type,string)
      string = 'Y-axis maximum.'
      call define (100,type,string)
      return
c
c      List the number of the transient recorder used to
c      measure each parameter here.
c
70    write(*,'('' Transient recorder associated with each ''
     c           ''parameter.'')')
      type = 'i'
      string = 'Cathode voltage.'
      call define (101,type,string)
      string = 'Intermediate anode voltage.'
      call define (102,type,string)
      string = 'Cathode current.'
      call define (103,type,string)
      string = 'Collector current.'
      call define (104,type,string)
```

```
      string = 'Microwave diode.'
      call define (105,type,string)
      string = 'Interferometer diode.'
      call define (106,type,string)
      return
c
c      List calibration factors for the DT2801 here.
c
80    write(*,'('' Calibration factors for the Data Translation''
     c     '' card only.'')')
      type = 'e'
      string = 'Supercon top coil current (A/V)'
      call define (111,type,string)
      string = 'Supercon bottom coil current (A/V)'
      call define (112,type,string)
      string = 'Trim top coil current (A/V)'
      call define (113,type,string)
      string = 'Trim bottom coil current (A/V)'
      call define (114,type,string)
      string = 'Cathode voltage (kV/V)'
      call define (115,type,string)
      string = 'Intermediate anode voltage (kV/V)'
      call define (116,type,string)
      string = 'Cathode current (A/V)'
      call define (117,type,string)
      string = 'Collector current (A/V)'
      call define (118,type,string)
      string = 'Microwave diode (kW/V)'
      call define (119,type,string)
      string = 'Calorimeter (W/V)'
      call define (120,type,string)
      string = 'Interferometer position (mills/V)'
      call define (121,type,string)
      string = 'Interferometer diode (W/V)'
      call define (122,type,string)
      return
c
c     List calibration factors for transient recorders.
c
90    write(*,'('' Calibration factors for the transient recorder''
     c            '' data only.'')')
      type = 'e'
      string = 'Cathode voltage (kV/V)'
      call define (131,type,string)
      string = 'Intermediate anode voltage (kV/V)'
      call define (132,type,string)
      string = 'Cathode current (A/V)'
      call define (133,type,string)
      string = 'Collector current (A/V)'
      call define (134,type,string)
      string = 'Microwave diode (kW/V)'
      call define (135,type,string)
      string = 'Interferometer diode (kW/V)'
      call define (136,type,string)
```

```
      return                                                       50
c
c      List attenuator values here.
c
100      write(*,'('' Attenuator number''10x''Value'')')
      do 105 i=1,7
105      write(*,'(7x,i3,14x,1pe12.5)')i,atten(i)
      return
c
c      List Data Translation data (datran(i,j)) here.
c
110      write(*,'('' Channel''10x''Mean''10x''Mean square'')')
      do 115 i=1,16
115      write(*,'(4x,i2,7x,1pe12.5,12x,1pe12.5)')i,(datran(i,j),
     c     j=1,2)
      return
c
c      List transient recorder data here (trrec(i,j)).
c
120      istrt = 1
      iend = 9
      if (cvar(2) .gt. 0.0 .and. cvar(2) .le. 5.0) then
      istrt = 2.0 * cvar(2) - 1.0
      iend = 2.0 * cvar(2) - 1.0
      endif
      do 125 i=istrt,iend,2
      num = (i+1) / 2
      write(*,'('' Transient recorder number '',i2)') num
      write(*,'(/'' Point''10x''Mean''10x''Mean square'')')
      do 125 j=1,int(uvar(17+num))
125   write(*,'(1x,i4,6x,1pe12.5,6x,1pe12.5)')j,(trrec(j,k),
     c    k=i,(i+1))
      RETURN
      END
```

```
C***********************************************************************
C
C    user3.qo
c
c    This module was written to interface PICAX with
c    the Quasi-optical gyrotron experiment at the
c    Naval Research Lab, Washington D.C.
c
c    Written by:
c              Tom Hargreaves
c              JAYCOR
c              205 S. Whiting St.
c              Alexandria, VA 22304
c
c    Last modification:        September 26, 1985
C
C***********************************************************************
c
c
$storage:2
c
C
      SUBROUTINE QUANAL
      write (*,100)
100   format (' Analyze the data'/
     c    ' This routine will calculate the peak power and the peak'/
     c    ' efficiency using the calorimeter data, the microwave diode'/
     c    ' data (if available) and the current values of the user'/
     c    ' variables.'//' ex:  an m'/
     c    ' where:       m = 0   will prompt for the rep rate'/
     c    '                = rep rate')
      RETURN
      END
c
C-----------------------------------------------------------------------
      SUBROUTINE QUPLOT
      write (*,100)
100   format (' Plot the data'/' ex: "pl n" will plot:'/' n'5x'plot'/
     1    ' 0      Cathode voltage vs. time'/
     2    ' 1      Intermediate voltage vs. time'/
     3    ' 2      Cathode current vs. time'/
     4    ' 3      Collector current vs. time'/
     5    ' 4      Microwave diode vs. time'/
     6    ' 5      Interferometer diode vs. time'/
     7    ' 6      Interferometer diode vs. mirror spacing'/
     8    ' 7      Microwave diode vs. cathode voltage'/
     9    ' 8      Microwave diode vs. cathode current'/
     1    ' 9      Microwave diode vs. collector current'/
     2    ' 10     Microwave diode vs. magnetic field')
c
      RETURN
      END
c
C-----------------------------------------------------------------------
```

```
      SUBROUTINE QUPROC
      write (*,100)
100   format (' Proceed'/' This command will restart the experiment'
     1   ' from the point where it was interupted.')
c
      RETURN
      END
c
C-----------------------------------------------------------------
      SUBROUTINE QUREAD
      write (*,100)
100   format (' Read a record from a disk file'/' ex: "r n m"'/
     1   ' will read record number m from logical unit number n.'/
     2   ' m = 0 will read the next record.'/
     3   ' n = 0 will read from the default LUN (3).')
c
      RETURN
      END
c
C-----------------------------------------------------------------
      SUBROUTINE QUSTRT
      write (*,100)
100   format (' Start the experiment'/' ex: "s n m"'/
     1   ' n = 0      will take the normal data.'/
     2   ' n = 1      will scan the interferometer.'/
     3   ' m = 0      will subtract the baseline from the transient '
     4   'recorder data.'/
     5   ' m >< 0     will not subtract the baseline.')
c
      RETURN
      END
c
C-----------------------------------------------------------------
      SUBROUTINE QUWRIT
      write (*,100)
100   format (' Write a record to a disk file'/' ex: "w n m"'/
     1   ' will write record number m to logical unit number n.'/
     2   ' m = 0 will write the next record.'/
     3   ' n = 0 will write to the default LUN (2).')
c
      RETURN
      END
c
C-----------------------------------------------------------------
      SUBROUTINE QUZERO
      write (*,100)
100   format (' Zero'/' will zero the accumulative data arrays.')
c
      RETURN
      END
```

```
c*********************************************************************
c      user4.go
c
c      This module was written to interface PICAX with
c      the Quasi-optical gyrotron experiment at the
c      Naval Research Lab, Washington D.C.
c
c      Written by:
c            Tom Hargreaves
c            JAYCOR
c            205 S. Whiting St.
c            Alexandria, VA 22304
c
c      Last modification:      October 1, 1985
c
c*********************************************************************
c
c
$storage:2
c
c
      subroutine sc b set (b,taper,func)
c
c---------------------------------------------------------------------
c
c      This routine services the superconducting magnet.
c
c      func = 0      Set the magnet currents.
c      func = 1      Check to see if the magnet has charged
c                  to full current yet. Set:
c                  func = 0      if done.
c                  func = 1      if not yet done.
c
c---------------------------------------------------------------------
c
c    See the addendum to the users manual for the derivation of
c    the formulae for the two current values.
c    This routine makes use of the following facts:
c
c    1) For a level field at 50 kG:
c              top coil current    = 118.8 amps
c              bottom coil current = 121.4 amps
c
c    2) The power supply level is programmed as:
c              0 volts =>   0 amps
c             -6 volts => 150 amps
c
c    3) The minutes full scale is still programmed at the power supply.
c       See section III, figure 5a in the IGC magnet manual.
c
c    4) The Transiac 3016 DAC has a resolution of + or - 3 millivolts
c       which corresponds to + or - 75 milliamps at the magnet.
```

```
c
c-----------------------------------------------------------------
c
      common / data / atten (20), datran (16,2), trrec (1000,10),
     c          freq (1000,2), data (100,10)
      COMMON /U VAR   / N U VAR, U VAR (150)
c
      if (func .eq. 0.0) then
c
c     Calculate the voltages to send to the DAC.
c
        top cur = (-b) * (taper * 147.45 - 10545.0) / (taper * 22.19 +
     c        4438.0)
        bot cur = b * (taper * 258.43 + 10775.0) / (taper * 22.19 +
     c        4438.0)
        top vol = (-top cur) * 6.0 / 150.0
        bot vol = (-bot cur) * 6.0 / 150.0
c
c     Ensure that the two voltage values are between 0 and -4.88 volts.
c
        if (top vol .gt. 0) top vol = 0.0
        if (top vol .lt. -4.88) top vol = -4.88
        if (bot vol .gt. 0) bot vol = 0.0
        if (bot vol .lt. -4.88) bot vol = -4.88
        n d top = nint (top vol * 3276.7)
        n d bot = nint (bot vol * 3276.7)
c
c     Get the camac slot number and the DAC channel numbers.
c
        n = int (uvar (64))
        n a top = int (uvar (47))
        n a bot = int (uvar (48))
        nq = 0
        nx = 0
        call camo (n, 16, n a top, n d top, nq, nx)
        if (nq .ne. 1) write (*,'('' Q = ''i4''  while attempting ''
     c      ''to set the top superconductor current.'')') nq
        nq = 0
        nx = 0
        call camo (n, 16, n a bot, n d bot, nq, nx)
        if (nq .ne. 1) write (*,'('' Q = ''i4''  while attempting ''
     c      ''to set the bottom superconductor current.'')') nq
c
c     Set up for DMA again
c
        call dmaset (1,2,1,1000)
        return
      elseif ( func .eq. 1.0) then
c
c     Compare the measured current to the desired current.
c
        if (abs (top cur - datran (int (uvar (31)),1)) .le. 0.1) then
         if (abs (bot cur - datran (int (uvar (32)),1)) .le. 0.1) then
         func = 0.0
```

```
        return
        endif
       endif
      endif
c
      return
      end
c
      subroutine tr b set (alpha,func)
c
c-----------------------------------------------------------------------
c
c      This routine services the trim coil.
c
c      func = 0      Set the magnet currents.
c      func = 1      Check to see if the magnet is fully
c                 charged yet. Set
c                 func = 0      if done.
c                 func = 1      if not yet done.
c
c-----------------------------------------------------------------------
c
      func = 0.0
      return
      end
c
      subroutine v set (volt,func)
c
c-----------------------------------------------------------------------
c
c      This routine sets the electron gun voltage.
c
c      func = 0      Set the voltage.
c      func = 1      Check to see if the modulator is fully
c                 charged yet. Set:
c                 func = 0      if done.
c                 func = 1      if not yet done.
c
c-----------------------------------------------------------------------
c
      func = 0.0
      return
      end
c
      subroutine cur set (cur,func)
c
c-----------------------------------------------------------------------
c
c      This routine sets the electron gun current.
c
c      func = 0      Set the current.
c      func = 1      Check to see if the gun is up to the
c                 specified current yet. Set:
c                 func = 0      if done.
```

```
c                       func = 1      if not yet done.
c
c-------------------------------------------------------------------
c
      func = 0.0
      return
      end
c
      subroutine gt attn (att)
c
c-------------------------------------------------------------------
c
c      This routine gets the attenuator values from the
c      Camac crate. It also gets the differential amplifier
c      value. The values are stored in array att(20) :
c
c      1-6       Lecroy 8102 hex attenuator.
c      7       Transiac differential amplifier.
c
c-------------------------------------------------------------------
c
      COMMON /U VAR   / N U VAR, U VAR (150)
      dimension att (20)
      integer d, q, x
      do 10 i=1,20
10    att(i) = 0.0
c
c      Set hex attenuator slot number to n.
c
      n = uvar (65)
      do 20 i = 1,6
      x = 0
      d = 0
      q = 0
      call cami (n,0,i,d,q,x)
      if(x .ne. 1) then
      write(*,'('' Error reading attenuator number ''i3)')i
      go to 20
      endif
      att (i) = 1.0/float(((d*5) / 4))
20    continue
c
c      Set differential amplifier slot number to n.
c
      n = uvar (66)
      x = 0
      d = 0
      q = 0
      call cami (n,0,0,d,q,x)
      if(x .ne. 1) then
      write(*,'('' Error reading differential amplifier'')')
      return
      endif
c
```

```
c      If d > 64 then filter is in.
c
      if (d .gt. 64) d = d - 128
      att (7) = 0.05
      if (d .gt. 0) att (7) = float (d) / 10.0
      if (d .gt. 2) att (7) = float (d) / 8.0
      if (d .gt. 16) att (7) = float (d) / 6.4
c
c    Set up for DMA again
c
      call dmaset (1,2,1,1000)
      return
      end
c
      subroutine real dt
c
c-----------------------------------------------------------------------
c
c      This routine converts the raw data from the Data
c      Translation card into real world units.
c
c      Note that the data in array dat (i,2) is squared.
c
c-----------------------------------------------------------------------
c
      common / data / atten (20), datran (16,2), trrec (1000,10),
     c          freq (1000,2), data (100,10)
      COMMON /U VAR   / N U VAR, U VAR (150)
c
      fac = 20.0 / 4096.0
      do 100 i = 1,16
c
c    Find the parameter associated with this data.
c
      n = 0
      do 10 j = 31,42
      if (i .eq. int (uvar(j))) n = j - 30
10    continue
c
c    Now get the calibration factor.
c
      cal = 1.0
      if (uvar (110+n) .ne. 0.0) cal = uvar (110+n)
      if (n .eq. 0) cal = 1.0
c
c    Now convert the data.
c
      datran (i,..   cal * (datran (i,1) * fac - 10.0)
100   datran (i,2) = (cal*(sqrt(datran (i,2)) * fac - 10.0)) ** 2
c
      return
      end
c
      subroutine real tr
```

```
c
c-----------------------------------------------------------------------
c
c      This routine converts the raw data from the Transiac
c      trasient recorders into real world units.
c
c      Note that the data in trrec (i,j),j=2,4,... is squared.
c
c      base fl = .true.                Subtract baseline.
c              = .false.               Do not subtract baseline.
c
c-----------------------------------------------------------------------
c
      common / data / atten (20), datran (16,2), trrec (1000,10),
     c          freq (1000,2), data (100,10)
      COMMON /U VAR   / N U VAR, U VAR (150)
      common / p c var / n c var, c var (4)
      common / u flags / strt fl, freq fl, base fl, n dt err, dt err
      logical strt fl, freq fl, base fl, dt err
c
c    This routine is set up for a maximum of 5 transient recorders.
c
      do 100 j = 1,5
      if (int(uvar(17+j)) .le. 0) go to 100
c
      n = int (uvar(66+j))
      nx = 0
      nq = 0
      nd = 0
c
c    Get the attenuator value and the calibration factor here.
c
      att = 1.0
      cal = 1.0
      do 10 i = 1,5
      if (int(uvar(100+i)) .eq. j) then
       if (int(uvar(80+i)) .ne. 0) att = atten (int(uvar(80+i)))
       if (uvar (130+i) .ne. 0.0) cal = uvar (130+i)
      endif
10    continue
      if (int(uvar(106)) .eq. j) then
       if (int(uvar(88)) .ne. 0) att = atten (int(uvar(88)))
       if (uvar (136) .ne. 0.0) cal = uvar (136)
      endif
c
c    Put the variance of the data (sigma**2) into tr rec (i,2*j).
c    sigma**2 = <x**2> - <x>**2
c
      do 20 i = 1,int(uvar(17+j))
20    tr rec (i,2*j) = tr rec (i,2*j) - (tr rec (i,2*j-1)**2)
c
c    If base fl is false, then don't get the offset data.
c
      if (.not. base fl) go to 45
```

```
c
c     Get the data offset from the transient recorder
c     and add it to the data.
c
c
c     In order to read the offset correctly, there must be no trigger
c     pulse until at least 700 microseconds after the F(27)*A(0)
c     camac call. Therefore, start this sequence immediately after
c     a trigger pulse. This method should be good for rep rates up
c     to at least 100 Hz.
c
      do 25 i=1,2
23    call camo (n,8,0,nd,nq,nx)
      if (nq .ne. 1) go to 23
      call camo (n,9,0,nd,nq,nx)
      if (nq .ne. 1) write (*,'('' Q = ''i4''  while attempting ''
     c    ''to start sampling for transient recorder ''i2)') nq, j
      call camo (n,26,0,nd,nq,nx)
      if (nq .ne. 1) write (*,'('' Q = ''i4''  while enabling the ''
     c    ''LAM for transient recorder ''i2)') nq, j
25    continue
      call camo (n,27,0,nd,nq,nx)
      if (nq .ne. 1) write (*,'('' Q = ''i4''  while attempting ''
     c    ''to read offset for transient recorder ''i2)') nq, j
c
c     Test the LAM to see if the transient recorder data
c     is ready to be read by the computer.
c
30    call camo (n,8,0,nd,nq,nx)
      if (nq .ne. 1) go to 30
c
c     Read the data here.
c
      do 40 i = 1,int(uvar(17+j))
      call cami (n,2,0,nd,nq,nx)
      tr rec (i,2*j-1) = tr rec (i,2*j-1) - float (nd)
      if (nq .eq. 0) write (*,'('' Q = 0 for transient recorder ''
     c   ''data point ''i2'' , ''i4)') i,j
40    continue
45    continue
c
c     Restart the transient recorder sampling and set
c     up the LAM again.
c
      call camo (n,9,0,nd,nq,nx)
      if (nq .ne. 1) write (*,'('' Q = ''i4''  while attempting ''
     c    ''to start sampling for transient recorder ''i2)') nq, j
      call camo (n,26,0,nd,nq,nx)
      if (nq .ne. 1) write (*,'('' Q = ''i4''  while enabling the ''
     c    ''LAM for transient recorder ''i2)') nq, j
c
c     Convert the data here.
c
      do 50 i = 1,int (uvar(17+j))
```

```
      tr rec (i,2*j-1) = cal * tr rec (i,2*j-1) * 0.002 / att
50    tr rec (i,2*j) = (cal *(sqrt (tr rec(i,2*j))) * 0.002 / att) ** 2
100   contimue
c
c    Set up for DMA again
c
      call dmaset (1,2,1,1000)
c
      return
      end
```

```
C*********************************************************************
C
c     user5.qo
c
c     This module was written to interface PICAX with
c     the Quasi-optical gyrotron experiment at the
c     Naval Research Lab, Washington D.C.
c
c     Written by:
c           Tom Hargreaves
c           JAYCOR
c           205 S. Whiting St.
c           Alexandria, VA 22304
c
c     Last modification:      September 27, 1985
C
C*********************************************************************
c
c
$storage:2
c
C
      SUBROUTINE UPDATE
C
c---------------------------------------------------------------------
c
c     This routine takes the data from the experiment.
c     If freq fl is true then only the interferometer
c     data is taken.
c
c---------------------------------------------------------------------
c
      COMMON /P FLAGS/ PROG ON, EXPT ON, QUERY
      common / p c var / n c var, c var (4)
      COMMON /U VAR   / N U VAR, U VAR (150)
      common / data / atten (20), datran (16,2), trrec (1000,10),
     c         freq (1000,2), data (100,10)
      common / stp num / n stp b, n stp tp, n stp v, n stp cr,
     c         n stp al, n dat, n freq, n dat ar
      common / increm / del b, del tp, del v, del cr, del al
      common / u flags / strt fl, freq fl, base fl, n dt err, dt err
      logical strt fl, freq fl, base fl, dt err
      LOGICAL          PROG ON, EXPT ON, QUERY
c
c     Get the data from the Data Translation board.
c
      call gt dtrn
c
c    Check for an error reading the DT2801.
c
      if (dt err) return
      if (.not. freq fl) then
c
c     Get the data from the Camac crate.
```

```
c
      call gt trec
      endif
c
c      Check the number of data shots taken so far.
c
      n dat = n dat + 1
      if (n dat .lt. int(uvar(17))) return
c
c      Average the data here.
c
      do 100 i=1,16
      do 100 j=1,2
100   datran (i,j) = datran (i,j) / uvar (17)
      if (.not. freq fl) then
      do 110 j=1,5
      do 110 i=1,int(uvar(17+j))
      trrec (i,2*j-1) = trrec (i,2*j-1) / uvar (17)
110   trrec (i,2*j) = trrec (i,2*j) / uvar (17)
      endif
c
c      Convert the data into real world units.
c
      call real dt
c
c     If freq fl is true then put the interferometer
c     data into the array freq.
c
      if (freq fl) then
      freq (nfreq,1) = datran (int(uvar(41)),1)
      freq (nfreq,2) = datran (int(uvar(42)),2)
      n freq = n freq + 1
      if (n freq .lt. int(uvar(16))) return
      n dat = 0
      else
      call real tr
      endif
c
c     Put data into the accumulative data arrays.
c
      n dat ar = n dat ar + 1
      data (n dat ar,1) = datran (int(uvar(39)),1)
      data (n dat ar,2) = datran (int(uvar(39)),2)
      data (n dat ar,3) = datran (int(uvar(35)),1)
      data (n dat ar,4) = datran (int(uvar(35)),1)
      data (n dat ar,5) = datran (int(uvar(37)),1)
      data (n dat ar,6) = datran (int(uvar(37)),1)
      data (n dat ar,7) = datran (int(uvar(38)),1)
      data (n dat ar,8) = datran (int(uvar(38)),1)
      data (n dat ar,9) = 25.0 * (datran (int(uvar(31)),1) / 118.8
     c    + datran (int(uvar(32)),1) / 121.4)
      data (n dat ar,10) = 25.0 * (datran (int(uvar(31)),2) / 118.8
     c    + datran (int(uvar(32)),2) / 121.4)
c
```

```
c    The experiment is finished.
c
      expt on = .false.
c
      RETURN
      END
c
C------------------------------------------------------------------
c
      SUBROUTINE UPROCD
c
c------------------------------------------------------------------
c
c      This routine restarts the experiment after it has
c      been interupted.
c
c      If:
c      n dat >= n dat max      then:      Increment the parameters,
c                              zero the data arrays, and
c                              return.
c      n dat < n dat max      then:      return.
c
c------------------------------------------------------------------
c
      common /p flags/ prog on, expt on, query
      common / p c var / n c var, c var (4)
      COMMON /U VAR   / N U VAR, U VAR (150)
      common / data / atten (20), datran (16,2), trrec (1000,10),
     c         freq (1000,2), data (100,10)
      common / stp num / n stp b, n stp tp, n stp v, n stp cr,
     c         n stp al, n dat, n freq, n dat ar
      common / increm / del b, del tp, del v, del cr, del al
      common / u flags / strt fl, freq fl, base fl, n dt err, dt err
      logical strt fl, freq fl, base fl, dt err
      logical prog on, expt on, query
      common / datime / idate (4), itime (4)
c
c      Check to see if the experiment has been started yet.
c
      if (.not. strtfl) then
      write(*,'('' The experiment has not been started yet.'')')
      return
      endif
      expt on = .true.
      if(ndat .lt. int(uvar(17))) go to 2000
c
c     Get the current time and date.
c
      call qtime (itime(1),itime(2),itime(3),itime(4))
      call qdate (idate(1),idate(2),idate(3))
c
c      Set n dat = 0, zero the data arrays, increment the
c      parameters, check the parameters, get the new (?)
c      attenuator values, and return.
```

```
c
      n dat = 0
      do 10 i=1,20
10    atten (i) = 0.0
      do 20 i=1,16
      do 20 j=1,2
20    datran (i,j) = 0.0
      do 30 i=1,1000
      do 30 j=1,10
30    trrec (i,j) = 0.0
      n freq = 1
      do 40 i = 1,1000
      do 40 j = 1,2
40    freq (i,j) = 0.0
c
c      Do the voltage first.
c
      n stp v = n stp v + 1
      if(n stp v .gt. int(uvar(9))) n stp v = 1
      volt = uvar(7) + delv*float(n stp v - 1)
      call v set (volt,0)
      if(n stp v .ne. 1)go to 95
c
c      Do the current next.
c
      n stp cr = n stp cr + 1
      if(n stp cr .gt. int(uvar(12))) n stp cr = 1
      cur = uvar(10) + del cr*float(n stp cr - 1)
      call cur set (cur,0)
      if(n stp cr .ne. 1)go to 95
c
c      Do alpha next.
c
      n stp al = n stp al + 1
      if(n stp al .gt. int(uvar(15))) n stp al = 1
      al = uvar(13) + del al*float(n stp al - 1)
      call tr b set (al,0)
      if (n stp al .ne. 1) go to 95
c
c      Do the taper next.
c
      n stp tp = n stp tp + 1
      if(n stp tp .gt. int(uvar(6))) n stp tp = 1
      tp = uvar(4) + del tp*float(n stp tp - 1)
c
c      Check to see if the magnetic field needs to
c      be incremented also.
c
      if(n stp tp .eq. 1) then
      n stp b = n stp b + 1
      if(n stp b .gt. int(uvar(3))) then
      expt on = .false.
      write (*,'('' Experiment complete.''/'' *''$)')
      return
```

```
      endif
      endif
      b = uvar(1) + del b*float(n stp b - 1)
      call sc b set (b,tp,0)
95    continue
c
c      Get the attenuator values here.
c
      call gt attn (atten)
      n dt err = 0
99    test nm = 0.0
100   continue
      test nm = test nm + 1.0
      if (test nm .ge. 10.0) then
       write(*,'('' System not yet ready.''/'' Enter 1 to bypass ''
     c        ''checks, 0 to continue checking.'')')
       read (*,*) test
       if (test .eq. 1.0) go to 1000
       go to 99
      endif
      call dt strt
110   call gtdtrn
      if (dt err) go to 110
      call realdt
      test = 1.0
      call sc b set (0,0,test)
      if (test .ne. 0.0) go to 100
      test = 1.0
      call tr b set (0,test)
      if (test .ne. 0.0) go to 100
      test = 1.0
      call v set (0,test)
      if (test .ne. 0.0) go to 100
      test = 1.0
      call cur set (0,test)
      if (test .ne. 0.0) go to 100
1000  continue
c
c     Zero the data array datran (16,2) again.
c
      do 1010 i=1,16
      do 1010 j=1,2
1010  datran (i,j) = 0.0
2000  continue
c
c     Wait for a trigger on the Data Translation board,
c     then restart the data sampling.
c
      call dt strt
c
c     Restart the transient recorder sampling and set
c     up the LAM again.
c
      do 3000 i = 1,5
```

```
      if (uvar (17+i) .eq. 0.0) go to 3000
      n = int (uvar (66+i))
      nd = 0
      nq = 0
      nx = 0
      call camo (n,9,0,nd,nq,nx)
      if (nq .ne. 1) write (*,'('' Q = ''i4''  while attempting ''
     c    ''to start sampling for transient recorder ''i2)') nq, i
      call camo (n,26,0,nd,nq,nx)
      if (nq .ne. 1) write (*,'('' Q = ''i4''  while enabling the ''
     c    ''LAM for transient recorder ''i2)') nq, i
3000  continue
c
      RETURN
      END
c
C-----------------------------------------------------------------------
c
      SUBROUTINE USTART
c
c-----------------------------------------------------------------------
c
c      This routine sets the experimental parameters (i.e. magnetic
c      field, etc.) in preparation for data taking. The actual data
c      taking is done in subroutine update. The data arrays are also
c      zeroed in this routine.
c
c      cvar (1) = 1.0    =>     Take interferometer data.
c
c      cvar (2) = 0.0    =>     Subtract baseline from transient
c                               recorder data.
c               >< 0.0   =>     Do not subtract baseline.
c
c-----------------------------------------------------------------------
c
      common / p c var / n c var, c var (4)
      COMMON /U VAR   / N U VAR, U VAR (150)
      common / data / atten (20), datran (16,2), trrec (1000,10),
     c         freq (1000,2), data (100,10)
      common / stp num / n stp b, n stp tp, n stp v, n stp cr,
     c         n stp al, n dat, n freq, n dat ar
      common / increm / del b, del tp, del v, del cr, del al
      common / u flags / strt fl, freq fl, base fl, n dt err, dt err
      logical strt fl, freq fl, base fl, dt err
      common / datime / idate (4), itime (4)
c
c     Get the current time and date.
c
      call qtime (itime(1),itime(2),itime(3),itime(4))
      call qdate (idate(1),idate(2),idate(3))
c
c      Initialize the data arrays here.
c
      do 10 i = 1,16
```

```
      do 10 j = 1,2
10    datran (i,j) = 0.0
      do 20 i = 1,1000
      do 20 j = 1,10
20    trrec (i,j) = 0.0
      freq fl = .false.
      do 30 i = 1,1000
      do 30 j = 1,2
30    freq (i,j) = 0.0
      do 40 i = 1,100
      do 40 j = 1,10
40    data (i,j) = 0.0
c
c      Set the experimental parameters here.
c
      call sc b set (uvar(1),uvar(4),0)
      call tr b set (uvar(13),0)
      call v set (uvar(7),0)
      call cur set (uvar(10),0)
c
c      Initialize the step numbers here.
c
      n dat ar = 0
      n stp b = 1
      n stp tp = 1
      n stp v = 1
      n stp cr = 1
      n stp al = 1
c
c      Calculate the increments here.
c
      del b = 0.0
      if (uvar(3) .gt. 1.0) then
      del b = (uvar(2) - uvar(1)) / (uvar(3) - 1.0)
      endif
      del tp = 0.0
      if (uvar(6) .gt. 1.0) then
      del tp = (uvar(5) - uvar(4)) / (uvar(6) - 1.0)
      endif
      del v = 0.0
      if (uvar(9) .gt. 1.0) then
      del v = (uvar(8) - uvar(7)) / (uvar(9) - 1.0)
      endif
      del cr = 0.0
      if (uvar(12) .gt. 1.0) then
      del cr = (uvar(11) - uvar(10)) / (uvar(12) - 1.0)
      endif
      del al = 0.0
      if (uvar(15) .gt. 1.0) then
      del al = (uvar(14) - uvar(13)) / (uvar(15) - 1.0)
      endif
c
c      Check to see if the parameters have been set yet.
c      i.e. is the supercon magnet up to field yet?
```

```
c                                                                          68
c
c      Get the attenuator values here.
c                                                                          59
      call gt attn (atten)
      n dt err = 0
95    test nm = 0.0
100   continue
      test nm = test nm + 1
      if (test nm .ge. 10.0) then
       write(*,'('' System not ready yet.''/'' Enter 1 to bypass''
     c        '' checks, 0 to continue checking.'')')
       read (*,*) test
       if (test .eq. 1.0) go to 1000
       go to 95
      endif
      call dt strt
110   call gtdtrn
      if (dt err) go to 110
      call realdt
      test = 1.0
      call sc b set (0,0,test)
      if (test .ne. 0.0) go to 100
      test = 1.0
      call tr b set (0,test)
      if (test .ne. 0.0) go to 100
      test = 1.0
      call v set (0,test)
      if (test .ne. 0.0) go to 100
      test = 1.0
      call cur set (0,test)
      if (test .ne. 0.0) go to 100
1000  continue
c
c    Zero the data array datran (16,2) again.
c
      do 1010 i=1,16
      do 1010 j=1,2
1010  datran (i,j) = 0.0
c
c    Start the transient recorders sampling, enable the LAM,
c    and enable the computer readout.
c
c    Note:      1) Save no pre-trigger samples.
c               2) Sample interval = 33.3 nsec for Data Translation
c                  model 2008f.
c
      do 2000 i = 1,5
      nq = 0
      nx = 0
      nd = 3
      if (uvar(17+i) .le. 512.0) nd = 4
      if (uvar(17+i) .le. 256.0) nd = 5
      if (uvar(17+i) .le. 128.0) nd = 6
```

```
      if (uvar(17+i) .le. 64.0) nd = 7
      nd = nd * 64
      if (uvar(17+i) .ne. 0.0) then
      n = int (uvar(66+i))
      call camo (n,16,0,nd,nq,nx)
      if (nq .ne. 1) write (*,'('' Q = ''i4''  while attempting to ''
     c   ''initialize transient recorder ''i2)') nq, i
      endif
2000  continue
c
c      Set the final parameters.
c
      n dat = 0
      n dt err = 0
      strt fl = .true.
      freq fl = .false.
      if (cvar(1) .eq. 1.0) freq fl = .true.
      base fl = .true.
      if (cvar(2) .ne. 0.0) base fl = .false.
c
c     Wait for a trigger on the Data Translation board,
c     then restart the data sampling.
c
      call dt strt
c
c     Restart the transient recorder sampling and set
c     up the LAM again.
c
      do 3000 i = 1,5
      if (uvar (17+i) .eq. 0.0) go to 3000
      n = int (uvar (66+i))
      nd = 0
      nq = 0
      nx = 0
      call camo (n,9,0,nd,nq,nx)
      if (nq .ne. 1) write (*,'('' Q = ''i4''  while attempting ''
     c    ''to start sampling for transient recorder ''i2)') nq, i
      call camo (n,26,0,nd,nq,nx)
      if (nq .ne. 1) write (*,'('' Q = ''i4''  while enabling the ''
     c    ''LAM for transient recorder ''i2)') nq, i
3000  continue
c
      RETURN
      END
c
C-----------------------------------------------------------------
c
      subroutine dt strt
c
c-----------------------------------------------------------------
c
c     This routine returns after the second trigger pulse to the
c     Data Translation board.
c
```

```
c-----------------------------------------------------------------
c
      common / DT 2801 / base ad, com reg, stat rg, dat reg
      integer base ad, com reg, stat rg, dat reg
c
c    Stop the DT2801, read any junk data, and clear any errors.
c    Send the start read command and wait for a trigger.
c    Do this twice and then start the experiment on the third loop.
c
      do 100 i = 1,3
c
c    Stop the DT2801, read any junk data, and clear any errors.
c
       call out (com reg, #f)
c
       junk = inp (dat reg)
       call waitl (stat rg, 4, 0)
       call out (com reg, #1)
c
c    Send the start read command.
c
      call waitl (stat rg, 4, 0)
      call out (com reg, #8e)
      if (i .lt. 3) call waitl (stat rg, 5, 0)
100   continue
c
      return
      end
c
      subroutine gt dtrn
c
c-----------------------------------------------------------------
c
c    This routine gets the data from the Data Translation
c    board. The data is placed into the array datran (16,2)
c    by the assembly language routine input (datran). The data
c    is placed as follows:
c
c              datran (i,1):   data
c              datran (i,2):   square of data (for error analysis)
c
c-----------------------------------------------------------------
c
      common / data / atten (20), datran (16,2), trrec (1000,10),
     c         freq (1000,2), data (100,10)
      COMMON /U VAR   / N U VAR, U VAR (150)
      common / DT 2801 / base ad, com reg, stat rg, dat reg
      integer base ad, com reg, stat rg, dat reg
      common / u flags / strt fl, freq fl, base fl, n dt err, dt err
      logical strt fl, freq fl, base fl, dt err
      dimension dt temp (16,2)
c
      dt err = .false.
c
```

```
c    Get the input.
c
      call dtinpl (dt temp)
c
c    Check for any errors.
:
      call waitl (stat rg, #4, 0)
      istat = inp (stat rg)
      if (istat .ge. #80) then
      dt err = .true.
      n dt err = n dt err + 1
c
c    Stop the DT2801 and read any junk data.
c
      call out (com reg, #f)
      junk = inp (dat reg)
c
c    Send the read error register command.
c
      call waitl (stat rg, 4, 0)
      call out (com reg, 2)
c
c    Read the error register. Note that there are two bytes.
c
      call waitl (stat rg, 5, 0)
      ierrl = inp (dat reg)
      call waitl (stat rg, 5, 0)
      ierrh = inp (dat reg)
c
      if (ierrl .ne. 0 .or. ierrh .ne. 4) then
       write (*,'('' DT2801 error register low byte = ''i3)') ierrl
       write (*,'('' DT2801 error register high byte = ''i3)') ierrh
       pause 'This data point will be ignored. Hit return to continue.'
      endif
c
c    Compare the number of errors so far to 0.1 * (number of data
c    points).
c
      if (n dt err .gt. (1+int(0.1*uvar(17)))) then
       write(*,'('' There have been ''i3'' DT 2801 errors so far.'')')
     c    n dt err
       pause 'Hit return to reset the error count and continue.'
       n dt err = 0
      endif
c
c    Wait for a trigger on the Data Translation board,
c    then restart the data sampling.
c
      call dt strt
c
c    Restart the transient recorder sampling and set
c    up the LAM again.
c
      do 3000 i = 1,5
```

```
      if (uvar (17+i) .eq. 0.0) go to 3000
      n = int (uvar (66+i))
      nd = 0
      nq = 0
      nx = 0
      call camo (n,9,0,nd,nq,nx)
      if (nq .ne. 1) write (*,'('' Q = ''i4''  while attempting ''
     c    ''to start sampling for transient recorder ''i2)') nq, i
      call camo (n,26,0,nd,nq,nx)
      if (nq .ne. 1) write (*,'('' Q = ''i4''  while enabling the ''
     c    ''LAM for transient recorder ''i2)') nq, i
3000  continue
      else
      do 50 i = 1,16
      datran (i,1) = datran (i,1) + dt temp (i,1)
50    datran (i,2) = datran (i,2) + dt temp (i,2)
c
c    Send the start read command again.
c
      call waitl (stat rg, 4, 0)
      call out (com reg, #8e)
      endif
c
      return
      end
c
      subroutine gt trec
c
c-------------------------------------------------------------
c
c      This routine gets the data from the Transiac
c      transient recorders. The data is placed into the array
c      trrec (1000,10):
c
c      trrec (i,2*j-1),j=1-5      data
c      trrec (i,2*j)  ,j=1-5      square of data (for error analysis).
c
c-------------------------------------------------------------
c
      common / data / atten (20), datran (16,2), trrec (1000,10),
     c          freq (1000,2), data (100,10)
      COMMON /U VAR   / N U VAR, U VAR (150)
      dimension ndata (1000)
c
      do 100 j = 1,5
      if (uvar(17+j) .le. 0.0) go to 100
      n = int (uvar (66+j))
      nx = 0
      nq = 0
      nd = 0
      ne = 0
c
c    Test the LAM to see if the transient recorder data
c    is ready to be read by the computer.
```

```
c
10     call camo (n,8,0,nd,nq,nx)
       if (nq .ne. 1) go to 10
c
c     Read the data here.
c
       call dmai (n,2,0,ndata(1),ne)
       if (ne .ne. 0) then
       write (*,'('' Error number ''i2'' while reading transient ''
      c    ''recorder number ''i2/'' No data was recorded.'')') ne, j
       go to 100
       endif
       do 20 i = 1,int(uvar(17+j))
       tr rec (i,2*j-1) = tr rec (i,2*j-1) + float (ndata(i))
20     tr rec (i,2*j) = tr rec (i,2*j) + float(ndata(i))*float(ndata(i))
c
c     Restart the transient recorder sampling and set
c     up the LAM again.
c
       call camo (n,9,0,nd,nq,nx)
       if (nq .ne. 1) write (*,'('' Q = ''i4''  while attempting ''
      c     ''to start sampling for transient recorder ''i2)') nq, i
       call camo (n,26,0,nd,nq,nx)
       if (nq .ne. 1) write (*,'('' Q = ''i4''  while enabling the ''
      c     ''LAM for transient recorder ''i2)') nq, i
100    continue
c
       return
       end
```

```
data        segment public 'data'
data        ends

dgroup      group   data
code        segment 'code'
            assume  cs:code, ds:dgroup, ss:dgroup

public      ittinr
ittinr      proc far
            push dx
            mov ah,6h
            mov dl,0ffh             ;don't check for  c
            int 21h                 ;get character from buffer
            cmp al,0h               ;al = 0 means no characters
            jz exit

            mov ah,6h               ;print the character
            mov dl,al
            int 21h

            cmp al,8                ;is this character a del?
            jnz exit                ;if not, exit

            mov dl,32               ;write space over deleted char.
            int 21h
            mov dl,8                ;move cursor back a space again
            int 21h

exit:       mov ah,0h
            pop dx
            ret
ittinr      endp
code        ends
end
```

```
;
;-----------------------------------------------------------------
;
;     Subroutine dtinpl (data)
;
;     This subroutine gets the data from the Data Translation
;     board and puts it into the data array
;     data (16,2). The data is in data (i,1) and the
;     square of the data is in data (i,2).
;
;     This routine is written to interface with the PICAX
;     program by:
;
;               Tom Hargreaves
;               JAYCOR
;               205 S. Whiting St.
;               Alexandria, VA   22304
;
;     Last update:      August 6, 1985
;
;-----------------------------------------------------------------
;
data            segment public 'data'
                base_ad dw 02ech
                com_reg dw 02edh
                stat_rg dw 02edh
                dat_reg dw 02ech
                mem     dw ?
data            ends

dgroup          group   data
code            segment 'code'
                assume  cs:code, ds:dgroup, ss:dgroup
public          dtinpl
dtinpl          proc far
                push bp
                mov bp,sp
      ; Get the input from the port first.
                les bx, dword ptr [bp+6]
                mov cx,16
input_loop:     push cx
                call wait2
                pop cx
                mov dx,dat_reg
                mov ax,0
                in al,dx              ;Get the low data byte.
                mov mem,ax            ;Store it in mem.
                push cx
                call wait2
                pop cx
                mov dx,dat_reg
                mov ax,0
                in al,dx              ;Get the high data byte.
                mov ah,al
```

```
                mov al,0
                add mem,ax              ;Total the data value in mem.
       ; Square the data value and store as floating point variables.
                fild  mem
                fld st[0]
                fmul st[1],st[0]
                fstp dword ptr es:[bx]
                fstp dword ptr es:[bx+64]
                fwait
                add bx,4
                loop input_loop
                mov sp,bp
                pop bp
                ret 04h
                dtinpl endp

wait2           proc near
                mov dx,stat_rg
                mov ax,0h
next_in:        in al,dx
                mov cx,5
                and cl,al
                mov ch,0
                cmp cx,0
                jle next_in
                ret
                wait2 endp

                code ends
end
```

```
data            segment public 'data'
data            ends

dgroup          group   data
code            segment 'code'
                assume  cs:code, ds:dgroup, ss:dgroup

public          waitl
waitl           proc far
                push bp
                mov bp,sp
                les bx, dword ptr[bp+14]
                mov dx,es:[bx]
                mov ax,0h
next_in:        in al,dx
                les bx,dword ptr[bp+6]
                mov cx,es:[bx]
                xor al,cl
                les bx,dword ptr[bp+10]
                mov cx,es:[bx]
                and cl,al
                mov ch,0
                cmp cx,0
                jle next_in
                mov sp,bp
                pop bp
                ret 0ch
                waitl endp
                code ends
end
```

```
data            segment public 'data'
data            ends

dgroup          group   data
code            segment 'code'
                assume  cs:code, ds:dgroup, ss:dgroup

public          inp
inp             proc far
                push bp
                mov bp,sp
                les bx, dword ptr[bp+6]
                mov dx,es:[bx]
                mov ax,0h
                in al,dx
                mov sp,bp
                pop bp
                ret 04h
                inp endp
                code ends
end
```

```
data        segment public 'data'
data        ends

dgroup      group   data
code        segment 'code'
            assume  cs:code, ds:dgroup, ss:dgroup

public      out
out         proc far
            push bp
            mov bp,sp
            les bx, dword ptr[bp+10]
            mov dx,es:[bx]
            les bx,dword ptr[bp+6]
            mov al,es:[bx]
            out dx,al
            mov sp,bp
            pop bp
            ret 08h
            out endp
            code ends
end
```

END

10-86

DTIC